Primer do Projeto de Documentação do FreeBSD para novos colaboradores

Primer do Projeto de Documentação do FreeBSD para novos colaboradores

Revision: 43184
2013-11-13 by hrs.

Abstract

Obrigado por tornar-se parte do Projeto de Documentação do FreeBSD. A sua contribuição é extremamente valiosa.

Este *Primer* do Projeto de Documentação do FreeBSD cobre tudo o que você precisa saber para começar a contribuir com o Projeto de Documentação do FreeBSD, desde as ferramentas e softwares que você estará utilizando (tanto os obrigatórios quanto os recomendados) à filosofia por trás do projeto de documentação.

Este documento é um trabalho em andamento, e não está completo. As sessões que sabemos estarem incompletas estão indicadas com um * no seu nome.

Important

ESTA DOCUMENTAÇÃO É FORNECIDA PELO PROJETO DE DOCUMENTAÇÃO DO FREEBSD "NO ESTADO" E QUAISQUER GARANTIAS EXPLÍCITAS OU IMPLÍCITAS, INCLUINDO, MAS NÃO LIMITADAS AS GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E A DE ADEQUAÇÃO PARA UMA FINALIDADE PARTICULAR SÃO DESMENTIDAS. EM NENHUM EVENTO O PROJETO DE DOCUMENTAÇÃO DO FREEBSD PODERÁ SER RESPONSABILIZADO POR QUAISQUER DANOS DIRETOS, INDIRETOS, INCIDENTAIS, ESPECIAIS, EXEMPLARES, OU CONSEQÜENTES (INCLUINDO, MAS NÃO LIMITADO A, OBTENÇÃO DE BENS OU SERVIÇOS SUBSTITUTOS; PERDA DE USO, DE DADOS, OU DE LUCROS; OU A INTERRUPÇÃO DE NEGÓCIOS) DE QUALQUER FORMA CAUSADO E EM QUALQUER TEORIA DE RESPONSABILIDADE, SE EM CONTRATO, EM RESPONSABILIDADE ESTRITA, OU PROCESSUAL (PASSÍVEL DE PROCESSO, INCLUINDO NEGLIGÊNCIA OU NÃO) LEVANTADA DE QUALQUER FORMA PELO USO DESTA DOCUMENTAÇÃO, MESMO QUE AVISADO DA POSSIBILIDADE DE TAIS DANOS.

English Version

This is an unofficial translation of the Standard FreeBSD Documentation Project Legal Notice into Brazilian Portuguese. It was not published by The FreeBSD Documentation Project, and does not legally state the distribution terms for documentation that uses the Standard FreeBSD Documentation Project Legal Notice -- only the original English text of the Standard FreeBSD Documentation Project Legal Notice does that. Therefore, the translated version should not be used as a valid legal notice. However, we hope that this translation will help Brazilian Portuguese speakers understand the Standard FreeBSD Documentation Project Legal Notice better.

Make sure to ALWAYS check the latest English version of the Standard FreeBSD Documentation Project Legal Notice in the English documentation version of the FreeBSD Handbook.

Redistribution and use in source (SGML DocBook) and "compiled" forms (SGML, HTML, PDF, PostScript, RTF and so forth) with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code (SGML DocBook) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

2. Redistributions in compiled form (transformed to other DTDs, converted to PDF, PostScript, RTF and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Important

THIS DOCUMENTATION IS PROVIDED BY THE FREEBSD DOCUMENTATION PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FREEBSD DOCUMENTATION PROJECT BE LIABLE FOR ANY DIRECT,

Table of Contents

List of Examples

Prefácio

1. Prompt do interpretador de comandos (shell)

A tabela seguinte mostra o *prompt* padrão do sistema e o *prompt* do super usuário. Os exemplos irão utilizar estes *prompts* para indicar com qual usuário o exemplo foi executado.

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

2. Convenções Tipográficas

A tabela seguinte descreve as convenções tipográficas utilizadas neste livro.

Propósito	Exemplos
Nome de um comando.	Utilize `ls -a` para listar todos os arquivos.
Nome de um arquivo.	Edite o seu arquivo `.login`.
Saída (*output*) de um programa na tela do computador.	`Você tem email.`
O que você digita, quando contrastado com a saída (*output*) do programa na tela do computador.	`% su` `Password:`
Referência a uma página de manual.	Utilize o su(1) para assumir outro nome de usuário.
Nome de usuário e de grupos de usuários	Apenas o `root` pode fazer isso.
Ênfase	Você *deve* fazer isso.
Variáveis da linha de comando; Substitua com o nome real ou com a variável.	Para deletar um arquivo, digite `rm nome_do_arquivo`
Variáveis de ambiente	O `$HOME` é o seu diretório `home`.

3. Notas, dicas, informações importantes, avisos e exemplos

Ao longo do texto aparecerão notas, avisos e exemplos.

Note

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

Tip

Dicas são representadas desta forma, e contêm informações que você pode achar úteis, ou que mostram uma maneira mais fácil de fazer alguma coisa.

Important

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

Warning

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

Example 1. Uma amostra de exemplo

Os exemplos são representados deste modo, e normalmente contêm exemplos que você deve analisar, ou então, mostram como deveriam ser 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.

Chapter 1. Visão Geral

Seja bem vindo ao Projeto de Documentação do FreeBSD. Documentação de boa qualidade é muito importante para o sucesso do FreeBSD, e o Projeto de Documentação do FreeBSD (FDP) é a origem de muitos destes documentos. Suas contribuições são muito importantes.

A finalidade principal deste documento é explicar claramente *como o FDP é organizado*, *como escrever e como submeter documentos para o FDP* , e *como utilizar de forma efetiva as ferramentas que estão disponíveis para você enquanto estiver escrevendo*.

Todos são bem vindos para se juntar ao FDP. Não existe nenhum requisito mínimo para a sua associação, nenhuma quota de documentos que você precise produzir por mês. Tudo o que você precisa fazer é se inscrever na lista de discussão do projeto de documentação do FreeBSD.

Depois que você tiver terminado de ler este documento, você deve:

- Saber quais documentos são mantidos pelo FDP.
- Ser capaz de ler e entender o código fonte SGML de um documento mantido pelo FDP.
- Ser capaz de efetuar alterações num documento.
- Ser capaz de enviar suas alterações de volta para revisão e eventual inclusão na documentação do FreeBSD.

1.1. Conjunto de Documentação do FreeBSD

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

Páginas de Manual
: As páginas de manual do sistema na língua inglesa não são escritas pelo FDP, porque elas são parte da base do sistema. Entretanto, o FDP pode (e tem) reescrever partes das páginas de manual existentes para torná-las mais claras, ou para corrigir imprecisões.

  Os times de tradução são responsáveis por traduzir as páginas de manual do sistema para diferentes idiomas. Estas traduções são mantidas pelo FDP.

FAQ
: O objetivo do FAQ é consolidar (no formato de perguntas e respostas curtas) as perguntas que foram feitas, ou que podem ser feitas, nas várias listas de discussão e newsgroups dedicados ao FreeBSD. O formato não permite respostas longas e detalhadas.

Handbook
: O Handbook almeja ser um compreensivo recurso de referência online para os usuários do FreeBSD.

Web Site
: Esta é a principal presença do FreeBSD na `World Wide Web`, visível em http://www.FreeBSD.org/ e em muitos sites espelhos ao redor do mundo. O web site é o primeiro contato de muitas pessoas com o FreeBSD.

A documentação do web site, do Handbook do FreeBSD e do FAQ estão disponíveis no repositório Subversion `doc/`, que está localizado em `svn://svn.FreeBSD.org/doc/`.

As páginas de manual estão disponíveis no repositório Subversion `src/`, que está disponível em `svn://svn.FreeBSD.org/base/`.

Isto significa que os logs das alterações realizadas nestes arquivos é visível para qualquer um, e qualquer pessoa pode utilizar o svn para ver as alterações.

Em adição, muitas pessoas escreveram tutoriais ou outros web sites relacionados ao FreeBSD. Alguns destes trabalhos também estão armazenados no repositório Subversion (com a permissão do autor). Em outros casos o autor decidiu manter o seu documento separado do repositório principal do FreeBSD. O FDP se esforça tanto quanto possível para fornecer os links para estes documentos.

1.2. Antes de você iniciar

Este documento assume que você já sabe:

- Como manter uma cópia local atualizada da documentação do FreeBSD, através da manutenção de uma cópia local do repositório Subversion do FreeBSD utilizando o svn.
- Como obter e instalar um novo software utilizando o sistema de ports ou o pkg_add(1) do FreeBSD.

1.3. Início Rápido

Se você deseja ir começando, e se sente seguro de que pode ir pegando as coisas à medida que avança, siga estas instruções.

1. Instale o meta-port textproc/docproj.

   ```
   # cd /usr/ports/textproc/docproj
   # make JADETEX=no install
   ```

2. Obtenha uma cópia local da árvore de documentação do FreeBSD (`doc`) utilizando o svn.

   Se a velocidade da sua conexão ou se o espaço de armazenamento do seu disco local forem motivo de preocupação, o mínimo que você vai precisar será uma cópia de trabalho dos diretórios `head/share`, e `head/idioma/share`. Por exemplo:

```
% mkdir -p head/share
% mkdir -p head/en_US.ISO8859-1/share
% svn checkout svn://svn.FreeBSD.org/doc/head/share head/share
% svn checkout svn://svn.FreeBSD.org/doc/head/en_US.ISO8859-1/
share head/en_US.ISO8859-1/share
```

Se você tiver abundância de espaço em disco, você pode retirar uma cópia de trabalho completa (de todos os subdiretórios da árvore doc).

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

3. Se você está preparando uma alteração de um artigo ou livro existente, retire uma versão de trabalho do arquivo do repositório. Se você está planejando contribuir com um novo livro ou artigo, então utilize um dos existentes como guia.

   Por exemplo, se você deseja contribuir com um novo artigo sobre como configurar uma VPN entre o FreeBSD e o Windows 2000, você pode fazer o seguinte:

   1. Retire uma cópia de trabalho do diretório `articles`.

      ```
      % svn checkout svn://svn.FreeBSD.org/doc/head/en_US.ISO8859-1/
      articles
      ```

   2. Copie um artigo existente para utilizar como template. Neste caso, você decidiu que o seu novo artigo iria para um diretório chamado `vpn-w2k`.

      ```
      % cd head/en_US.ISO8859-1/articles
      % svn export committers-guide vpn-w2k
      ```

   Se você deseja editar um documento existente, como por exemplo o FAQ, o qual está em `head/en_US.ISO8859-1/books/faq`, você deve retirar a cópia de trabalho do repositório da seguinte forma:

   ```
   % svn checkout svn://svn.FreeBSD.org/doc/head/en_US.ISO8859-1/
   books/faq
   ```

4. Edite os arquivos `.xml` utilizando o editor da sua preferência.

5. Teste a marcação SGML utilizando o alvo `lint` com o comando make. Isto irá listar rapidamente qualquer erro existente no documento sem realizar qualquer tipo de transformação no seu arquivo, o que consumiria tempo.

   ```
   % make lint
   ```

   Quando você estiver pronto para efetivamente compilar o documento, você pode especificar um único formato ou uma lista de formatos de destino, na variável `FORMATS`. Atualmente os formatos suportados são, `html`, `html-split`, `txt`, `ps`, `pdf`, e `rtf`. A lista mais atualizada dos formatos suportados está listada no início do arquivo `head/`

`share/mk/doc.docbook.mk` . Certifique-se de utilizar aspas (") em volta da lista de formatos quando você estiver compilando mais de um formato num único comando.

Por exemplo, para converter o documento apenas para `html`, você deve utilizar:

```
% make FORMATS=html
```

Mas quando você deseja converter o documento tanto para o formato `html` quanto para o formato `txt`, você pode utilizar duas execuções separadas do make(1), como a seguir:

```
% make FORMATS=html
% make FORMATS=txt
```

ou, você pode fazer isso em um único comando:

```
% make FORMATS="html txt"
```

6. Envie suas alterações utilizando o send-pr(1).

Chapter 2. Ferramentas

O FDP utiliza diferentes ferramentas como auxílio no gerenciamento da documentação do FreeBSD, e na conversão para diferentes formatos, e assim por diante. Você irá precisar utilizar estas ferramentas se for trabalhar com a documentação do FreeBSD.

Todas estas ferramentas estão disponíveis como `Ports` e `Packages` do FreeBSD, simplificando enormemente o seu trabalho para instalá-las.

Você precisará instalar estas ferramentas antes de trabalhar com qualquer exemplo dos próximos capítulos. O uso real destas ferramentas será abordado nos próximos capítulos.

Se possível use o textproc/docproj

Você pode economizar bastante tempo se instalar o `port` textproc/docproj. Este é um *meta-port* que por si só não contém nenhum programa. Ao invés disto, ele depende que já estejam instalados corretamente vários outros `ports`. O processo de instalação *irá* baixar e instalar automaticamente todos os pacotes listados como necessários neste capítulo.

Um dos pacotes que você pode precisar é o conjunto de macros JadeTeX. No entanto, esse conjunto de macros requer que o TeX esteja instalado. O TeX é um pacote grande, e ele somente será necessário se você quiser gerar documentos nos formatos Postscript ou PDF.

Para economizar seu tempo e espaço em disco você deve especificar se quer, ou não, a instalação do JadeTeX (e por consequência do TeX) quando o `port` for instalado. Conforme necessário, faça:

```
# make JADETEX=yes install
```

ou

```
# make JADETEX=no install
```

Alternativamente você pode instalar o textproc/docproj-jadetex ou o textproc/docproj-nojadetex. Estes `ports` secundários irão definir a variável `JADETEX` para você, consequentemente eles irão instalar o mesmo conjunto de aplicativos na sua máquina. Observe que você poderá produzir apenas documentos em HTML e ASCII se você não instalar o JadeTeX. Para produzir documentos em PostScript e PDF você irá precisar do TeX.

2.1. Ferramentas Obrigatórias

2.1.1. Software

Estes programas são necessários para você trabalhar com a documentação do FreeBSD, e permitirão a conversão da mesma para os formatos HTML, texto puro e RTF. Eles estão todos incluídos em textproc/docproj.

Jade (textproc/jade)
: Uma implementação DSSSL. Utilizado para a conversão de documentos escritos com linguagem de marcas para outros formatos, incluindo HTML e TeX.

Tidy (www/tidy)
: Um HTML *"pretty printer"* , utilizado para reformatar alguns dos HTMLs gerados automaticamente ficando mais fácil de entendê-los.

Links (www/links)
: Um navegador WWW em modo texto que também converte arquivos HTML para texto puro.

peps (graphics/peps)
: Parte da documentação inclui imagens, algumas delas estão armazenadas como arquivos EPS. Estas imagens precisam ser convertidas para o formato PNG antes de serem exibidas em um navegador `web`.

2.1.2. Entidades e DTDs

Estes são os conjuntos de DTDs e de entidades usados pelo FDP. Eles precisam estar instalados para que você possa trabalhar com qualquer parte da documentação.

HTML DTD (textproc/html)
: HTML é a linguagem de marcas escolhida para a `World Wide Web` , e é usada no `web site` do FreeBSD.

DocBook DTD (textproc/docbook)
: DocBook é uma linguagem de marcas projetada para documentação técnica. Toda a documentação do FreeBSD está escrita em DocBook.

ISO 8879 entities (textproc/iso8879)
: 19 dos conjuntos de entidade de caracter ISO 8879:1986 utilizados por muitos DTDs. Inclui símbolos matemáticos nomeados, caracteres do conjunto de caracter Latin (acentos, diacríticos e assim por diante), e símbolos gregos.

2.1.3. Stylesheets

As `Stylesheets` são usadas na conversão e formatação de documentos para serem apresentados na tela, impressos, e assim por diante.

Modular DocBook Stylesheets (textproc/dsssl-docbook-modular)
: As `Modular DocBook Stylesheets` são usadas na conversão da documentação escrita em DocBook para outros formatos, tais como HTML ou RTF.

2.2. Ferramentas Opcionais

Você não precisa ter qualquer uma das ferramentas a seguir instaladas. Entretanto, você poderá achar mais fácil trabalhar com a documentação se elas estiverem disponíveis, elas também oferecem uma maior flexibilidade em relação aos formatos nos quais os documentos podem ser gerados.

2.2.1. Software

JadeTeX e teTeX (print/jadetex e print/teTeX)
: O Jade e o teTeX são usados para converter DocBook para os formatos DVI, Postscript, e PDF. As macros do JadeTeX são necessárias para estas conversões.

  Se você não pretende converter seus documentos para um destes formatos (i.e., HTML, texto puro, e RTF são o suficiente) então não será preciso instalar o JadeTeX e teTeX. Isto pode resultar em uma boa economia de tempo e espaço em disco, já que o teTeX possui tamanho de aproximadamente 30MB.

Important

Se você decidir instalar o JadeTeX e teTeX então será preciso configurar o teTeX depois do JadeTeX ter sido instalado. O arquivo `print/jadetex/pkg-message` contém instruções detalhadas sobre o que é preciso ser feito.

Emacs ou XEmacs (editors/emacs ou editors/xemacs)
: Ambos editores incluem um modo especial para a edição de documentos com uma linguagem de marcas que siga um SGML DTD. Esse modo inclui comandos para reduzir o volume total de digitação a ser feita, o que ajuda a reduzir a possibilidade de erros.

  Você não precisa utilizá-los, qualquer editor pode ser usado para editar documentos escritos com linguagem de marcas. Entretanto, se optar por usá-los você poderá constatar que eles tornam seu trabalho mais eficiente.

Se alguém tiver sugestões sobre algum outro `software` que seja útil para a manipulação de documentos SGML, por favor informe a Equipe de Engenharia de Documentação <doceng@FreeBSD.org >, desta forma ele poderá ser adicionado a esta lista.

Chapter 3. SGML Primer

A maioria dos documentos do FDP é escrita utilizando SGML. Este capítulo irá explicar exatamente o que isso significa, como ler e compreender os fontes dos documentos e os truques de SGML que você irá se defrontar na documentação.

Partes desta seção foram inspiradas no documento Começando a utilizar o DocBook de autoria do Mark Galassi.

3.1. Visão Geral

Antigamente, era simples de se lidar com um texto eletrônico. Naquela época, você tinha que saber em qual conjunto de caracteres o seu documento havia sido escrito (ASCII, EBCDIC ou um dos inúmeros outros), e mais nada. O texto era texto, e o que você via era realmente o que você tinha. Nenhuma frescura, nenhuma formatação, nenhuma inteligência.

Inevitavelmente, isto não era o suficiente. Uma vez que você tem o texto em uma máquina num formato utilizável, você espera que o equipamento seja capaz de usá-lo e manipulá-lo de forma inteligente. Você pode desejar indicar que uma determinada frase deve ser enfatizada, ou adicionada a um glossário, ou ser interligada a outra parte do documento. Você pode querer que os nomes dos arquivos sejam exibidos com uma fonte de estilo "typewriter" quando forem exibidos na tela, mas como "itálico " quando impresso, ou qualquer outra opção dentre a infinidade de opções disponíveis para apresentação.

Esperava-se que a inteligência artificial (AI) torna-se isso fácil. O seu computador leria o documento e identificaria automaticamente as frases chave, nomes de arquivos, os campos que o leitor teria que preencher, e muito mais. Infelizmente, a vida real não evoluiu como esperado, e os nossos computadores necessitam de algum auxílio antes que eles possam processar significativamente nosso texto.

Mais precisamente, eles precisam de ajuda para identificar o que é o que. Vejamos o texto abaixo:

> Para remover o `/tmp/foo` utilize rm(1).
>
> ```
> % rm /tmp/foo
> ```

Nele podemos facilmente visualizar quais partes são nomes de arquivos, quais são comandos que devem ser digitados, quais partes são referências às páginas de manual, etc. Mas o computador processando o documento não pode. Para isto nós precisamos utilizar uma marcação (`markup`).

A "marcação" é comumente utilizada para descrever a "adição de valor" ou o "aumento de custo". O termo (term) faz exame de ambos os meios quando aplicados ao texto. A

marcação é um texto adicional incluído no documento, e de alguma forma destacado do seu conteúdo, de modo que os programas que forem processá-lo possam ler as marcações e utilizá-las ao tomar decisões sobre o documento. Os editores podem ocultar a marcação do usuário, de forma que o usuário não se distraia com ele.

As informações extras armazenadas na marcação *adicionam valor* ao documento. Tipicamente a adição da marcação ao documento precisa ser realizada por uma pessoa — apesar de tudo, se os computadores pudessem reconhecer suficientemente bem o texto para adicionar as marcações, então não haveria necessidade de adicioná-las em primeiro lugar. Isto *aumenta o custo* (isto é, o esforço requerido) para criar o documento.

O exemplo acima foi na verdade escrito neste documento como se segue:

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

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

Como você pode ver, a marcação está claramente separada do conteúdo.

Obviamente, se você estiver iniciando no uso de marcações, você precisa definir o que a sua marcação significa, e como ela será interpretada. Você vai precisar de uma linguagem de marcação a qual você possa seguir quando estiver marcando os seus documentos.

Naturalmente, uma linguagem de marcação pode não ser o bastante. Os requisitos de uma linguagem de marcação destinada formatação de documentos técnicos são diferentes dos requisitos de uma linguagem de marcação destinada a formatação de receitas culinárias. Esta, por sua vez, seria muito diferente de uma linguagem de marcação usada para formatar poemas. O que você realmente precisa é de uma linguagem primária, a qual você possa utilizar para escrever estas e outras linguagens de marcação. Uma *meta linguagem de marcação*.

É exatamente isso que a *Standard Generalized Markup Language* (SGML) é. Muitas linguagens de marcação foram escritas em SGML, incluindo as duas mais utilizadas pelo FDP, o HTML e o DocBook.

Cada definição de linguagem é mais corretamente chamada de Definição de Tipo de Documento (DTD). O DTD especifica o nome dos elementos que podem ser utilizados, em qual ordem eles aparecem (e se alguma marcação pode ser utilizada dentro de outra marcação) e as informações relacionadas. Um DTD é algumas vezes referenciado como uma *aplicação* do SGML.

Um DTD é uma especificação *completa* de todos os elementos que podem ser utilizados, da ordem em que podem aparecer, quais elementos são obrigatórios, quais são opcionais, e assim por diante. Isto torna possível escrever um interpretador (parser) SGML, que leia ambos os DTD e um documento que reividique se adequar ao DTD. O interpretador pode

então confirmar se todos os elementos obrigatórios do DTD estão (ou não) presentes no documento na ordem correta, e se existem erros na marcação. Isto é normalmente referenciado como "validação do documento".

Note

Este processamento simplesmente confirma se a escolha dos elementos, a sua ordenação, etc, estão de acordo com o especificado no DTD. Ele *não* verifica se você utilizou a marcação *adequada* para o conteúdo. Se você tentasse marcar todos os nomes de arquivo em seu documento como nomes de funções, o interpretador não iria apontar isto como um erro (assumindo, naturalmente, que a sua DTD define elementos para nomes de arquivos e para funções, e que eles podem ser utilizados nos mesmos lugares).

É provável que a maioria das suas contribuições ao projeto de documentação irão se constituir de conteúdos marcados tanto em HTML quanto em DocBook, em vez de alterações nos DTDs. Por esta razão este livro não irá abordar a criação de um DTD.

3.2. Elementos, tags, e atributos

Todos os DTDs escritos em SGML compartilham certas características. Isto é uma dura surpresa, como a filosofia por de trás do SGML nos mostrará ser completamente inevitável. Uma das manifestações mais óbvias desta filosofia está no *conteúdo* e nos *elementos*.

A sua documentação (independente se é uma única página web ou um livro longo) é composta de conteúdo. Este conteúdo é então dividido (e de novo subdividido) em elementos. O propósito da adição de marcações é atribuir nome e identidade para os limites destes elementos de forma a possibilitar o processamento adicional.

Por exemplo, considere um livro típico. No nível mais alto, o livro por si só é um elemento. Este elemento "livro" (book) obviamente contém capítulos, os quais também podem ser considerados elementos em sua própria forma. Cada capítulo irá conter mais elementos, tais como parágrafos, citações, notas de rodapé, etc. Cada parágrafo pode conter elementos adicionais, identificando o conteúdo que era de discurso direto, ou o nome de um personagem da história.

Você pode preferir pensar nisto como uma "quebra" do conteúdo. No nível mais alto você tem um pedaço, o Livro. Olhando um pouco mais abaixo, você tem mais pedaços, os capítulos individuais. Estes estão divididos em pedaços ainda menores, os parágrafos, notas de rodapé, nomes de personagens, etc.

Observe que você pode fazer esta diferenciação entre os diferentes elementos do conteúdo sem recorrer a nenhum termo SGML. Na realidade é surpreendentemente fácil de usar. Você pode fazer isso utilizando uma caneta de marcação e uma cópia impressa do livro, utilizando diferentes cores para indicar os diferentes pedaços do conteúdo.

Naturalmente, nós não possuímos uma caneta eletrônica de marcação, assim nós necessitamos de alguma outra maneira de indicar a que elemento cada peça de conteúdo pertence. Nas linguagens escritas em SGML (HTML, DocBook, etc) isto é feito através do uso de *tags*.

Uma tag é utilizada para identificar onde um elemento particular começa e onde ele termina. *A tag não é uma parte própria do elemento .* Porque cada DTD foi normalmente escrito para marcar um tipo específico de informação, cada um deles reconhecerá diferentes elementos, e terá nomes diferentes para cada tag.

Para um elemento chamado *`element-name`* a tag de início normalmente irá se parecer com *`element-name`*. E a tag correspondente de fechamento para este elemento seria */`element-name`*.

Example 3.1. Utilizando um elemento (tags de inicio e fim)

O HTML possui um elemento para indicar que o conteúdo envolvido por este elemento é um parágrafo, chamado p. Este elemento possui ambas as tags de início e de fim.

```
<p>Este é um parágrafo.  Ele inicia com a tag de inicio do
  elemento 'p', e irá terminar com a tag de fim para o
  elemento 'p'.</p>

<p>Este é um outro parágrafo.  Mas este é muito menor.</p>
```

Nem todos os elementos requerem uma tag de finalização. Alguns elementos não possuem conteúdo. Por exemplo, em HTML você pode indicar que deseja que uma linha horizontal apareça no documento. Obviamente, esta linha não possui conteúdo, assim apenas a tag de inicio é requerida para este elemento.

Example 3.2. Utilizando um elemento (Apenas tag de início)

O HTML possui um elemento para indicar uma linha horizontal, chamado `hr`. Este elemento não contém nenhum conteúdo, assim ele possui apenas uma tag de inicio.

```
<p>Este é um parágrafo.</p>

<hr>

<p>Este é outro parágrafo.  Uma linha horizontal o separa do
  parágrafo anterior.</p>
```

Se isto não é óbvio agora, os elementos podem conter outros elementos. No exemplo anterior do livro, o elemento livro continha todos os elementos capítulos, os quais por sua vez continham todos os elementos parágrafos, etc.

Example 3.3. Elementos contendo elementos; `em`

```
<p>Este é um <em>parágrafo</em> simples no qual
  algumas das <em>palavras</em> foram <em>enfatizadas</em>.</p>
```

O DTD irá especificar as regras detalhando quais elementos podem conter outros elementos, e o que exatamente eles podem conter.

Important

As pessoas sempre confundem os termos tags e elementos, e utilizam os termos como se eles fossem intercambiáveis. Eles não são.

Um elemento é uma parte conceitual do seu documento. Um elemento possui um inicio e fim determinados. As tags marcam onde os elementos começam e terminam.

Quando este documento (ou qualquer pessoa que conheça SGML) se refere a “tag `p`” estamos nos referindo literalmente ao texto de três

caracteres <, p e >. Mas a frase "o elemento p" se refere ao elemento inteiro.

Esta distinção *é* muito sutil. Mas mantenha ela em mente

Os elementos podem ter atributos. Um atributo possui um nome e um valor, e é utilizado para adicionar informações extras ao elemento. Esta pode ser a informação a qual indica como o conteúdo deve ser renderizado, ou pode ser algo que identifique a ocorrência única do elemento, ou pode ser qualquer outra coisa.

O atributo de um elemento é sempre escrito *dentro* da tag de início para aquele elemento, e assume a forma `nome-do-atributo="valor-do-atributo"` .

Nas versões suficientemente recentes do HTML, o elemento `p` possui um atributo chamado `align`, o qual sugere o alinhamento (Justificação) de um parágrafo para o programa que estiver exibindo o HTML.

O atributo `align` pode assumir um de quatro valores possíveis, `left` (esquerda), `center` (centralizado), `right` (direita) e `justify` (justificado). Se o atributo não for especificado será assumido o valor padrão `left`.

Example 3.4. Utilizando um elemento com um atributo

```
<p align="left">A inclusão de um atributo de alinhamento neste
  parágrafo foi supérfluo, uma vez que o alinhamento
  padrão é left (esquerda).</p>

<p align="center">Isto pode aparecer no centro.</p>
```

Alguns atributos irão assumir apenas valores específicos, como o `left` ou `justify`. Outros irão permitir que você entre com qualquer coisa que deseje. Se você precisar incluir aspas (`"`) no valor de um atributo, você deve envolver o valor do atributo com aspas simples (`'`).

Example 3.5. Aspas simples envolta de atributos

```
<p align='right'>Eu estou a direita!</p>
```

Algumas vezes você não precisa utilizar aspas em volta de todos os valores dos atributos. Entretanto, a regra para fazer isso é muito sutil, e é muito mais simples *sempre* utilizar as aspas em volta dos valores dos seus atributos.

A informação nos atributos, elementos e tags são armazenados nos catálogos SGML. Várias ferramentas do projeto de documentação utilizam estes arquivos de catalogo para validarem o seu trabalho. As ferramentas no textproc/docproj incluem uma variedade de arquivos de catalogo SGML. O projeto de documentação do FreeBSD inclui seu próprio conjunto de arquivos de catálogos. Suas ferramentas precisam reconhecer ambos os tipos de arquivos de catalogo.

3.2.1. Para você fazer...

Para poder praticar com os exemplos deste documento você precisará instalar alguns aplicativos no seu sistema, além de assegurar que as variáveis de ambiente estejam corretamente configuradas.

1. Faça o download e instale o textproc/docproj a partir do sistema de ports do FreeBSD. Ele é um *meta-port* o qual deve efetuar o download e a instalação de todos os aplicativos e arquivos de suporte que são utilizados pelo projeto de documentação.

2. Adicione linhas ao seu arquivo de inicialização do shell para configurar a variável de ambiente `SGML_CATALOG_FILES` . (Se você não estiver trabalhando com a versão no idioma inglês da documentação, você pode precisar substituir o caminho com o diretório correto para o seu idioma.)

Example 3.6. **`.profile`**, para os usuários dos shells sh(1) e bash(1)

```
SGML_ROOT=/usr/local/share/xml
SGML_CATALOG_FILES=${SGML_ROOT}/jade/catalog
SGML_CATALOG_FILES=${SGML_ROOT}/docbook/4.1/catalog:
$SGML_CATALOG_FILES
SGML_CATALOG_FILES=${SGML_ROOT}/html/catalog:
$SGML_CATALOG_FILES
SGML_CATALOG_FILES=${SGML_ROOT}/iso8879/catalog:
$SGML_CATALOG_FILES
SGML_CATALOG_FILES=/usr/doc/share/xml/catalog:
$SGML_CATALOG_FILES
SGML_CATALOG_FILES=/usr/doc/en_US.ISO8859-1/share/xml/
catalog:$SGML_CATALOG_FILES
export SGML_CATALOG_FILES
```

Example 3.7. **`.cshrc`**, para os usuários dos shell csh(1) e tcsh(1)

```
setenv SGML_ROOT /usr/local/share/xml
setenv SGML_CATALOG_FILES ${SGML_ROOT}/jade/catalog
setenv SGML_CATALOG_FILES ${SGML_ROOT}/docbook/4.1/catalog:
$SGML_CATALOG_FILES
setenv SGML_CATALOG_FILES=${SGML_ROOT}/html/catalog:
$SGML_CATALOG_FILES
setenv SGML_CATALOG_FILES=${SGML_ROOT}/iso8879/catalog:
$SGML_CATALOG_FILES
setenv SGML_CATALOG_FILES /usr/doc/share/xml/catalog:
$SGML_CATALOG_FILES
setenv SGML_CATALOG_FILES /usr/doc/en_US.ISO8859-1/share/
xml/catalog:$SGML_CATALOG_FILES
```

Para carregar estas variáveis, execute um logout do sistema, logando novamente em seguida, ou execute os comandos acima na sua linha de comando para configurar os valores das variáveis.

1. Crie o arquivo `example.xml` , e entre com o seguinte texto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
  <head>
    <title>Um exemplo de arquivo HTML</title>
  </head>

  <body>
    <p>Este é um parágrafo contendo algum texto.</p>

    <p>Este parágrafo contém mais algum texto.</p>

    <p align="right">Este parágrafo pode estar alinhado a ↺
direita.</p>
  </body>
</html>
```

2. Tente validar este arquivo utilizando um interpretador SGML.

   Um dos componentes do textproc/docproj é o `onsgmls`, um interpretador de validação [10]. Normalmente, o `onsgmls` lê um documento marcado de acordo com um DTD SGML e retorna uma cópia do conjunto de informações sobre a estrutura dos elementos (ESIS, mas isso não é importante agora).

Entretanto, quando o `onsgmls` é executado com o parâmetro `-s`, ele irá suprimir o output normal, e imprimir apenas as mensagens de erro. Isto o torna um meio útil de verificar se o seu documento é válido ou não.

Utilize o `onsgmls` para verificar se o seu documento é válido:

```
% onsgmls -s example.xml
```

Como você irá ver, o `onsgmls` irá executar sem retornar nenhuma mensagem. Isto significa que o seu documento foi validado com sucesso.

3. Veja o que acontece quando um elemento obrigatório é omitido. Tente remover as tags `title` e `/title` , e execute novamente a validação.

```
% onsgmls -s example.xml
onsgmls:example.xml:5:4:E: character data is not allowed here
onsgmls:example.xml:6:8:E: end tag for "HEAD" which is not ↺
finished
```

As mensagens de erro emitidas pelo `onsgmls` são organizadas em grupos separados por dois pontos, ou colunas.

Coluna	Propósito
1	O nome do programa que está gerando o erro. Ela será sempre `onsgmls`.
2	O nome do arquivo que contém o erro.
3	Número da linha na qual o erro aparece.
4	Número da coluna na qual o erro aparece.
5	Um código de uma letra indicando a natureza da mensagem. `I` indica uma mensagem informativa, `W` é para um aviso, e `E` é para um erro [a], e `X` é para uma referência cruzada. Como você pode ver, estas mensagens são erros.
6	O texto da mensagem.

[a] Ele não está sempre na quinta coluna. O `onsgmls -sv` exibe `onsgmls:I: "OpenSP" version "1.5.2"` (depende da versão instalada). Como você pode ver, esta é uma mensagem informativa.

A simples omissão das tags `title` gerou 2 erros diferentes.

O primeiro erro indica que o conteúdo (neste caso, caracteres, ou melhor, a tag de inicio de um elemento) ocorreu onde o interpretador SGML estava esperando outra

coisa. Neste caso, o interpretador estava esperando encontrar uma das tags de início para os elementos que são válidos dentro do `head` (como a `title`).

O segundo erro é porque o elemento `head` *deve* conter o elemento `title`. Por causa disso o `onsgmls` considera que o elemento não foi corretamente finalizado. Entretanto, a tag de finalização indica que o elemento foi fechado antes que estivesse terminado.

4. Coloque de volta o elemento `title`.

3.3. A declaração DOCTYPE

O inicio de cada documento que você escrever deve especificar o nome do DTD o qual se aplica ao seu documento. Isto deve ser feito para que os interpretadores SGML possam determinar o DTD e assegurar que o documento esta em conformidade com o mesmo.

Esta informação é geralmente expressada em uma linha, na declaração DOCTYPE.

Uma declaração típica para um documento escrito para conformar-se com a versão 4.0 do DTD HTML se parece com esta:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN">
```

Esta linha contém diferentes componentes.

`<!`
: É o *indicador* que indica que se trata de uma declaração SGML. Esta linha está declarando o tipo do documento.

`DOCTYPE`
: Mostra que esta é uma declaração SGML para o tipo de documento.

`html`
: O nome do primeiro elemento o qual irá aparecer no documento

`PUBLIC "-//W3C//DTD HTML 4.0//EN"`
: Lista o Identificador Público Formal (FPI) para o DTD ao qual este documento conforma-se. O seu interpretador SGML irá utiliza-lo para encontrar o DTD correto quando estiver processando o documento.

  O `PUBLIC` não faz parte do FPI, ele indica para o processador SGML como localizar o DTD referenciado na FPI. Outras formas de informar ao interpretador SGML como localizar o DTD serão abordadas mais tarde .

`>`
: Retorno ao documento.

3.3.1. Identificadores públicos formais (FPIs)

Note

Você não precisa conhecê-los, mas eles são um background útil, e podem ajudá-lo a depurar problemas quando o seu processador SGML não puder localizar o DTD que você esta utilizando.

Os FPIs devem seguir uma sintaxe específica. Esta sintaxe é a seguinte:

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

Proprietário
: Isto indica o proprietário da FPI.

  Se este conjunto de caracteres começar com "ISO" significará que o FPI é de propriedade do ISO. Por exemplo, a FPI `"ISO 8879:1986//ENTITIES Greek Symbols//EN"` lista o `ISO 8879:1986` como sendo o proprietário do conjunto de entidades dos símbolos Gregos. O ISO 8879:1986 é o numero da ISO para o padrão SGML.

  De outra forma, este conjunto de caracteres irá se parecer com `-// Proprietário` ou `+//Proprietário` (Observe que a única diferença é a introdução do + ou do -).

  Se o conjunto de caracteres começar com - significa que o proprietário da informação não é registrado, se começar com um + significa que ele é registrado.

  O ISO 9070:1991 define como os nomes registrados são gerados; ele pode ser derivado do numero de uma publicação ISO, de um código ISBN, ou um código de organização atribuído de acordo com o ISO 6523. Além disso, uma autoridade de registro pode ser criada a fim de atribuir nomes registrados. O conselho ISO delegou isto ao `American National Standards Institute` (ANSI).

  Como o Projeto FreeBSD não foi registrado o conjunto de caracteres de proprietário é `-//FreeBSD`. E como você pode ver, o W3C também não é um proprietário registrado.

Palavra-Chave
: Existem diversas palavras-chave as quais indicam o tipo de informação no arquivo. Algumas das palavras chaves mais comuns são `DTD`, `ELEMENT`, `ENTITIES`, e `TEXT`. A palavra chave `DTD` é utilizada apenas para os arquivos DTD, a `ELEMENT` é normalmente utilizada para fragmentos DTD os quais contenham apenas entidades e declarações de elementos. A palavra `TEXT` é utilizada para o conteúdo SGML (texto e tags).

Descrição
: Qualquer descrição que você deseje fornecer para o conteúdo deste arquivo. O que pode incluir um número de versão ou qualquer texto curto o qual seja significativo para você e único para o sistema SGML.

Idioma
: Este é um código ISO de duas letras o qual identifica o idioma nativo do arquivo. O código `EN` é utilizado para o idioma inglês.

3.3.1.1. Arquivos de `catálogo`

Se você utilizar a sintaxe acima e processar este documento utilizando um processador SGML, o processador irá precisar de uma forma de associar a FPI ao nome do arquivo no seu computador o qual contém o DTD.

Para isto devemos utilizar um arquivo de catálogo. Um arquivo de catálogo (tipicamente chamado de `catalog`) contém linhas as quais mapeiam FPIs para nomes de arquivos. Por exemplo, se o arquivo de catálogo contiver a linha:

```
PUBLIC "-//W3C//DTD HTML 4.0//EN" "4.0/strict.dtd"
```

O processador SGML saberia que deveria procurar pelo DTD `strict.dtd` no subdiretório `4.0` de qualquer diretório que possuísse um arquivo `catalog` contendo esta linha.

Veja o conteúdo do `/usr/local/share/xml/html/catalog` . Este é o arquivo de catálogo para o DTD HTML o qual será instalado como parte do port textproc/docproj.

3.3.1.2. `SGML_CATALOG_FILES`

A fim de encontrar um arquivo de `catálogo`, o seu processador SGML precisará saber onde procurar. Muitos deles possuem recursos de parâmetros de linha de comando para especificar o caminho para um ou mais catálogos.

Adicionalmente, você pode definir a variável de ambiente `SGML_CATALOG_FILES` para apontar para os arquivos. Esta variável deve consistir de uma lista, separada por dois pontos (":"), de arquivos de catálogo (incluindo seus caminhos completos).

Tipicamente, você precisará incluir os seguintes arquivos:

- `/usr/local/share/xml/docbook/4.1/catalog`
- `/usr/local/share/xml/html/catalog`
- `/usr/local/share/xml/iso8879/catalog`
- `/usr/local/share/xml/jade/catalog`

Você já deve ter feito isto.

3.3.2. Alternativas aos FPIs

Ao invés de utilizar um FPI para indicar o DTD ao qual o documento conforma-se (e consequentemente, quais arquivos no sistema contém o DTD), você pode especificar explicitamente o nome do arquivo.

A sintaxe para isto é ligeiramente diferente:

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

A palavra chave `SYSTEM` indica que o processador SGML deve encontrar o DTD em um local específico do sistema. Isto tipicamente (mas não sempre) significa que o DTD será fornecido como um nome de arquivo.

O uso de FPIs é preferido por razões de portabilidade. Você pode não desejar ter que enviar uma cópia do DTD junto com seu documento, e se você utilizasse um identificador `SYSTEM` todos necessitariam manter os seus DTDs no mesmo lugar que você.

3.4. Voltando para o SGML

Como mencionado anteriormente, o SGML é utilizado somente quando escrevemos um DTD. Isto não é estritamente verdade. Existem certas sintaxes SGML as quais você poderá desejar utilizar com os seus documentos. Por exemplo, você pode incluir comentários no seu documento, e eles serão ignorados pelo interpretador. Os comentários são adicionados utilizando sintaxe SGML. Outros usos para a sintaxe SGML no seu documento serão mostrados mais tarde.

Obviamente, você precisa indicar de alguma forma ao processador SGML que o conteúdo seguinte não se trata de elementos do documento, mas sim de SGML sobre o qual o interpretador deve atuar.

Estas sessões são marcadas no seu documento com `<! ... >`. Tudo entre estes delimitadores é sintaxe SGML tal como você pode encontrar dentro de um DTD.

Como você pode perceber, a declaração DOCTYPE é um exemplo de sintaxe SGML a qual você precisa incluir no seu documento.

3.5. Comentários

Comentários são uma construção SGML, e são normalmente válidos apenas dentro de um DTD. Entretanto, como mostrou a Section 3.4, “Voltando para o SGML”, é possível utilizar sintaxe SGML com os seus documentos.

O delimitador para um comentário SGML é o conjunto de caracteres “`--`”. A primeira ocorrência deste conjunto de caracteres abre um comentário e a segunda fecha.

Example 3.8. Comentário SGML genérico

```
<!-- Teste de comentário -->
```

```
<!-- Este é o interior do comentário -->

<!-- Este é outro comentário    -->

<!-- Esta é uma forma
     de fazer comentários de várias linhas -->

<!-- Esta é outra forma  --
  -- de fazer comentários de várias linhas -->
```

Se você já utilizou HTML antes, você pode ter sido exposto a regras diferentes para comentários. Em particular, você pode pensar que o conjunto de caracteres `<!--` abre um comentário e que ele apenas é fechado por um `-->`.

Este *não* é o caso. Muitos dos navegadores web possuem interpretadores HTML quebrados, e irão aceitar isso como válido. Entretanto, os interpretadores SGML utilizados pelo projeto de documentação são muito mais rígidos, e irão rejeitar os documentos que contiverem este erro.

Example 3.9. Comentários SGML errados

```
<!-- Este é o comentário --

     Isto está fora do comentário!

  -- de volta para dentro do comentário -->
```

O interpretador SGML irá tratar isto como ele é realmente;

```
<!Isto está fora do comentário>
```

Isto não é um SGML válido, e pode dar mensagens de erro confusas.

```
<!----- Isto é uma idéia muito ruim ----->
```

E como o exemplo sugere, *não escreva* comentários como esse.

```
<!--================================================-->
```

Esta é uma abordagem (ligeiramente) melhor, mas ele ainda é potencialmente confuso para as pessoas novas no uso do SGML.

3.5.1. Para você fazer...

1. Adicione alguns comentários ao arquivo `example.xml` e verifique se ele continua válido usando o `onsgmls`.

2. Adicione alguns comentários inválidos ao `example.xml` e veja as mensagens de erro que o `onsgmls` emite quando encontra um comentário inválido.

3.6. Entidades

Entidades são um mecanismo para atribuir nomes para pedaços de conteúdo. Quando o interpretador SGML processar o seu documento, qualquer entidade que ele encontrar será substituída pelo conteúdo da entidade.

Esta é uma boa forma de ter pedaços de conteúdo reutilizáveis e facilmente alteráveis em seus documentos SGML. Esta é também a única forma de incluir um arquivo marcado dentro de outro utilizando SGML.

Existem dois tipos de entidades que podem ser utilizadas em duas situações diferentes; *Entidades gerais* e *Entidades de parâmetros*.

3.6.1. Entidades gerais

Você não pode utilizar uma entidade geral em um contexto SGML (embora você as defina em um). Elas podem ser utilizadas apenas no seu documento. Compare isto com as entidades de parâmetros.

Cada entidade geral possui um nome. Quando você quer referenciar uma entidade geral (e consequentemente incluir o texto que ela representa no seu documento), você escreve `&nome-da-entidade;`. Por exemplo, suponha que você possui uma entidade chamada `current.version`, a qual expande para a versão atual do seu produto. Você pode escrever:

```
<para>A versão atual do nosso produto é &amp;current.version;.</para>
```

Quando o número de versão mudar, você pode simplesmente alterar a definição do valor da entidade geral e reprocessar o seu documento.

Você também pode utilizar entidades gerais para incorporar caracteres que você não poderia incorporar de outra forma em um documento SGML. Por exemplo, os caracteres

< e & não podem aparecer normalmente em um documento SGML. Quando o interpretador SGML vê o símbolo < ele assume que aquilo é uma tag (uma tag de abertura ou de fechamento) que está a ponto de aparecer, e quando ele vê o símbolo & ele assume que o próximo texto será o nome de uma entidade.

Felizmente, você pode utilizar as duas entidades gerais `<` e `&` sempre que você precisar incluí-los.

Uma entidade geral só pode ser definida dentro de um contexto SGML. Tipicamente, isto é feito imediatamente depois da declaração DOCTYPE.

Example 3.10. Definindo uma entidade geral

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [
<!ENTITY current.version    "3.0-RELEASE">
<!ENTITY last.version       "2.2.7-RELEASE">
]>
```

Observe como a declaração DOCTYPE foi estendida adicionando-se um colchete no final da primeira linha. As duas entidades estão definidas nas próximas duas linhas, antes que o colchete seja fechado, e então a declaração DOTYPE é fechada.

O colchete é necessário para indicar que nós estamos extendendo o DTD indicado pela declaração DOCTYPE.

3.6.2. Entidades de parâmetro

Assim como as entidades gerais , as entidades de parâmetro são utilizadas para atribuir um nome a pedaços reutilizáveis de texto. Entretanto, enquanto as entidades gerais só podem ser utilizadas com o seu documento, as entidades de parâmetro podem ser utilizadas apenas dentro de um contexto SGML.

As entidades de parâmetro são definidas de uma forma similar as entidades gerais. Entretanto, ao invés de utilizar `&nome-da-entidade;` como referência, utiliza `%nome-da-entidade;` [1]. A definição também inclui o `%` entre a palavra chave `ENTITY` e o nome da entidade.

Example 3.11. Definindo entidades de parâmetros

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [
<!ENTITY % param.some "some">
```

[1] Entidades de Parâmetro utilizam o símbolo de Porcentagem.

```
<!ENTITY % param.text "text">
<!ENTITY % param.new  "%param.some more %param.text">
]>
```

Isto pode não parecer particularmente útil. Mas ele será.

3.6.3. Para você fazer...

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" [
<!ENTITY version "1.1">
]>

<html>
  <head>
    <title>Um exemplo de arquivo HTML</title>
  </head>

  <body>
    <p>Este é um parágrafo contendo algum texto.</p>

    <p>Este parágrafo contém mais algum texto.</p>

    <p align="right">Este parágrafo pode estar alinhado a ↺
direita.</p>

    <p>A versão atual deste documento é: &amp;version;</p>
  </body>
</html>
```

2. Valide o documento utilizando o `onsgmls`

3. Carregue o arquivo `example.xml` no seu navegador web (Você pode precisar copiá-lo para `example.html` antes que o seu navegador possa reconhecê-lo como um documento HTML).

   A menos que o seu navegador seja muito avançado, você não irá ver a entidade referenciada por `&version;` substituída pelo número de versão. A maioria dos navegadores web possuem interpretadores muito simplistas os quais não manuseiam corretamente SGML [2].

4. A solução é *normalizar* seu documento utilizando um normalizador SGML. Um normalizador lê um SGML válido e retorna um SGML igualmente válido o qual foi transformado de alguma forma. Uma das formas em que o normalizador transforma o

[2] Isto é uma vergonha. Imagine todos os problemas e `hacks` (tais como os Server Side Includes) que poderiam ser evitados se eles o fizessem.

SGML é expandindo todas as entidades referenciadas no documento, substituindo as entidades pelo texto que elas representam.

Você pode utilizar o `osgmlnorm` para fazer isto:

```
% osgmlnorm example.xml > example.html
```

Você deve encontrar uma cópia normalizada (isto é, entidades referenciadas expandidas) do seu documento no arquivo `example.html` , pronta para ser carregada no seu navegador web.

5. Se você examinar o retorno do `osgmlnorm` você ira ver que ele não inclui a declaração DOCTYPE no inicio. Para incluí-la você precisa utilizar a opção `-d`:

```
% osgmlnorm -d example.xml > example.html
```

3.7. Utilizando entidades para incluir arquivos

As entidades (ambas gerais e de parâmetros) são particularmente úteis quando utilizadas para incluir um arquivo dentro de outro.

3.7.1. Utilizando entidades gerais para incluir arquivos

Suponha que você possui algum conteúdo para um livro SGML organizado em arquivos, um arquivo por capítulo, chamados `chapter1.xml` , `chapter2.xml` , e assim por diante, com um arquivo `book.xml` que irá conter estes capítulos.

A fim de utilizar o conteúdo destes arquivos como os valores para as suas entidades, você as declara com a palavra chave `SYSTEM`. Isto direciona o interpretador SGML para utilizar o conteúdo dos arquivos nomeados como o valor da entidade.

Example 3.12. Utilizando entidades gerais para incluir arquivos

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [
<!ENTITY chapter.1 SYSTEM "chapter1.xml">
<!ENTITY chapter.2 SYSTEM "chapter2.xml">
<!ENTITY chapter.3 SYSTEM "chapter3.xml">
]>

<html>

  &chapter.1;
  &chapter.2;
  &chapter.3;
```

```
</html>
```

Warning

Quando utilizar uma entidade geral para incluir outros arquivos em um documento, os arquivos que estiverem sendo inclusos (`chapter1.xml` , `chapter2.xml` , etc) *não devem* iniciar com uma declaração DOCTYPE. Isto é um erro de sintaxe.

3.7.2. Utilizando entidades de parâmetro para incluir arquivos

Recorde-se que uma entidade de parâmetro só pode ser utilizada dentro de um contexto SGML. Por que então você desejaria incluir um arquivo dentro de um contexto do SGML?

Você pode utilizar isto para assegurar-se de que você pode reutilizar as suas entidades gerais.

Suponha que você possui muitos capítulos em seu documento, e você reutiliza estes capítulos em dois livros diferentes, cada livro organizando os capítulos de uma forma diferente.

Você pode listar as entidades no topo de cada livro, mas isto rapidamente torna-se incomodo de gerenciar.

Em vez de disso, coloque as definições das suas entidades gerais em um arquivo, e utilize uma entidade de parâmetro para incluir este arquivo dentro do seu documento.

Example 3.13. Utilizando entidades de parâmetro para incluir arquivos.

Primeiro, coloque as suas definições de entidades em um arquivo separado, chamado `chapters.ent` . Este arquivo contém o seguinte:

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

Agora crie uma entidade de parâmetro para referenciar o conteúdo do arquivo. E então utilize a entidade de parâmetro para carregar o arquivo no seu documento, o que tornará todas as entidades gerais disponíveis para uso. Feito isso, utilize as entidades gerais como antes;

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [
<!ENTITY % chapters SYSTEM "chapters.ent">
%chapters;
]>

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

3.7.3. Para você fazer...

3.7.3.1. Utilizando entidades gerais para incluir arquivos

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

   Coloque um conteúdo similar ao seguinte em cada arquivo:

   ```
   <p>Este é o primeiro parágrafo.</p>
   ```

2. Edite o arquivo `example.xml` para que ele se pareça com este:

   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [
   <!ENTITY version "1.1">
   <!ENTITY para1 SYSTEM "para1.xml">
   <!ENTITY para2 SYSTEM "para2.xml">
   <!ENTITY para3 SYSTEM "para3.xml">
   ]>

   <html>
     <head>
       <title>Um exemplo de arquivo HTML</title>
     </head>

     <body>
       <p>A versão atual deste documento é: &amp;version;</p>

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

3. Produza o arquivo `example.html` através da normalização do arquivo `example.xml` .

   ```
   % osgmlnorm -d example.xml > example.html
   ```

4. Carregue o arquivo `example.html` no seu navegador web, e confirme que os arquivos `paran.xml` foram incluídos no `example.html` .

3.7.3.2. Utilizando uma entidade de parâmetro para incluir arquivos.

Note

Você deve ter executado os passos anteriores primeiro.

1. Edite o arquivo `example.xml` para que ele se pareça com este:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [
<!ENTITY % entities SYSTEM "entities.xml"> %entities;
]>

<html>
  <head>
    <title>Um exemplo de arquivo HTML</title>
  </head>

  <body>
    <p>A versão atual deste documento é: &amp;version;</p>

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

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

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

3. Produza o arquivo `example.html` através da normalização do arquivo `example.xml` .

```
% osgmlnorm -d example.xml > example.html
```

4. Carregue o arquivo `example.html` no seu navegador web e confirme que os arquivos `paran.xml` foram incluídos no arquivo `example.html` .

3.8. Sessões marcadas

O SGML fornece um mecanismo para indicar que uma parte particular do documento deve ser processada de uma forma especial. Estas partes são denominadas “sessões marcadas”.

Example 3.14. Estrutura de uma sessão marcada

```
<![ Palavra-chave  [
  Conteúdo da sessão marcada
]]>
```

Como você esperaria, sendo uma construção SGML, uma sessão marcada inicia com um `<!`.

O primeiro colchete começa a limitar a sessão marcada.

A *`Palavra-chave`* descreve como esta sessão marcada deve ser processada pelo interpretador.

O segundo colchete indica que o conteúdo da sessão marcada inicia aqui.

A sessão marcada é finalizada pelo fechamento dos dois colchetes e então retornando ao contexto do documento a partir do contexto SGML com o `>`.

3.8.1. Palavras-Chave de sessões marcadas

3.8.1.1. `CDATA`, `RCDATA`

Estas palavras chave denotam o *modelo do conteúdo* das sessões marcadas, e permitem que você o altere a partir do padrão.

Quando um interpretador SGML esta processando um documento ele tenta seguir o que chamamos de "modelo de conteúdo"

Resumidamente, o modelo de conteúdo descreve que tipo de conteúdo o interpretador esta esperando encontrar, e o que fará com ele quando o encontrar.

Os dois modelos de conteúdo que você provavelmente irá achar mais úteis são o `CDATA` e o `RCDATA`.

O `CDATA` é para "Dados de Caracter". Se o interpretador está neste modelo de conteúdo então ele está esperando encontrar caracteres, e apenas caracteres. Neste modelo os símbolos `<` e o `&` perdem o seu status especial, e serão tratados como caracteres ordinários.

O `RCDATA` é para "Referências de entidade e dados de caracter ". Se o interpretador está neste modelo de conteúdo ele está esperando encontrar caracteres *e* entidades. O símbolo `<` perde o seu status especial, mas o `&` continuará sendo tratado como o inicio de uma entidade geral.

Isto é particularmente útil se você está incluindo algum texto literal o qual contém muitos caracteres < e &. Ao invés de atravessar o texto todo tendo que verificar se todos os < estão convertidos para um < e todos os & estão convertidos para um & pode ser mais simples marcar a sessão como contendo apenas CDATA. Quando o interpretador SGML encontrá-los ele irá ignorar os símbolos < e & embutidos no conteúdo.

Note

Quando você utiliza CDATA ou RCDATA nos exemplos de texto marcado em SGML, tenha em mente que o conteúdo do CDATA não é validado. Você tem que verificar o SGML incluso no texto utilizando algum outro meio. Você pode, por exemplo, escrever o exemplo em outro documento, validar o código de exemplo, e então colá-lo para o seu conteúdo CDATA.

Example 3.15. Utilizando uma sessão marcada como CDATA

```
<para>Aqui está um exemplo de como você incluiria algum texto ↺
que contenha muitos
  símbolos <literal><</literal> e <literal>></literal>.
  O texto de exemplo é um fragmento de HTML.
  O texto circunvizinho (<para> e <programlisting>) é do
  DocBook.</para>

<programlisting>
  <![CDATA[>![RCDATA[
    <p>Esta é uma amostra que apresenta alguns elementos de ↺
HTML.
       Uma vez que os símbolos de < e > são utilizados muitas ↺
vezes, é
       mais fácil dizer que o exemplo todo é uma sessão ↺
marcada do
       tipo CDATA, do que utilizar nomes de entidades para ↺
representar
       estes símbolos ao longo de todo o texto.</p>

    <ul>
      <li>Este é um item de lista</li>
      <li>Este é um segundo item de lista</li>
      <li>Este é um terceiro item de lista</li>
    </ul>

    <p>Este é o final do exemplo.</p>]]>
  -]]>
```

```
</programlisting>
```

Se você examinar o fonte deste documento você irá ver que esta técnica foi utilizada por toda parte.

3.8.1.2. `INCLUDE` and `IGNORE`

Se a palavra chave for `INCLUDE` então o conteúdo da sessão marcada será processado. Se a palavra chave for `IGNORE` então a sessão marcada será ignorada e não será processada. Ela não irá aparecer no output.

Example 3.16. Utilizando `INCLUDE` e `IGNORE` nas sessões marcadas

```
<![ INCLUDE [
  Este texto será processado e incluído.
]]>

<![ IGNORE [
  Este texto não será processado nem incluído.
]]>
```

Por si só, isto não é muito útil. Se você desejar remover o texto do seu documento você pode cortá-lo fora, ou comentá-lo.

Torna-se mais útil quando você utilizar entidades de parâmetro para controlá-lo. Lembre-se que entidades de parâmetro só podem ser utilizadas em um contexto SGML, e que a palavra chave de uma sessão marcada *é* um contexto SGML.

Por exemplo, suponha que você produza uma cópia impressa e uma versão eletrônica de alguma documentação. Você pode desejar incluir na versão eletrônica algum conteúdo extra, o qual não deve aparecer na versão impressa.

Crie uma entidade de parâmetro, e configure seu valor para `INCLUDE`. Escreva seu documento, usando uma sessão marcada para delimitar o conteúdo que deve aparecer apenas na versão eletrônica. Nesta sessão marcada utilize a entidade de parâmetro no lugar da palavra chave.

Quando você desejar produzir uma cópia impressa do documento, altere o valor da entidade de parâmetro para `IGNORE` e reprocesse o documento.

Example 3.17. Utilizando uma entidade de parâmetro para controlar uma sessão marcada

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [
<!ENTITY % electronic.copy "INCLUDE">
]]>

...

<![ %electronic.copy [
  Este conteúdo deve aparecer apenas na versão eletrônica do
  documento.
]]>
```

Quando for produzir uma versão impressa do documento, altere a definição da entidade para;

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

Ao reprocessar o documento, a sessão marcada que utilizar a entidade `%electronic.copy` como a sua palavra chave será ignorada.

3.8.2. Para você fazer...

1. Crie um novo arquivo chamado `section.xml` , o qual deve conter o seguinte conteúdo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [
<!ENTITY % text.output "INCLUDE">
]>

<html>
  <head>
    <title>Um exemplo utilizando uma sessão marcada.</title>
  </head>

  <body>
    <p>Este parágrafo <![CDATA[contêm muitos caracteres <
      (< < < < <) de forma que é mais simples utilizar
      uma sessão marcada do tipo CDATA.-]]></p>

    <![ IGNORE [
    <p>Este parágrafo definitivamente não será incluído
      no output.</p>
    -]]>
```

```
    <![ %text.output [
    <p>Este parágrafo pode ou não aparecer no output.</p>

    <p>A sua ocorrência é controlada pela entidade de parâmetro ↺
%text.output
      .</p>
    -]]>
  </body>
</html>
```

2. Normalize este arquivo utilizando o `osgmlnorm` e examine o resultado. Observe quais parágrafos apareceram, quais desapareceram e o que aconteceu com o conteúdo da sessão marcada como CDATA.

3. Altere a definição da entidade `text.output` de `INCLUDE` para `IGNORE`. Re-normalize o arquivo, e observe o resultado para ver o que foi alterado.

3.9. Conclusão

Esta é a conclusão da introdução ao SGML. Devido a restrições de espaço e complexidade, diversos assuntos não foram abordados em profundidade (ou sequer foram abordados). Entretanto, as sessões prévias cobriram o suficiente do SGML para que você possa seguir a organização da documentação do FDP.

Chapter 4. Marcação SGML

Esse capítulo descreve as duas linguagens de marcação que você vai encontrar quando for contribuir para o projeto de documentação do FreeBSD. Cada seção descreve a linguagem de marcação, e detalha a marcação que você provavelmente vai querer usar ou que já está utilizando.

Estas linguagens de marcação contém um grande número de elementos, e as vezes pode ser confuso escolher qual elemento usar em uma situação específica. Esta seção apresenta os elementos que provavelmente você vai precisar e fornece exemplos de como utilizá-los.

Esta *não* é uma lista detalhada de elementos, uma vez que ela apenas ratifica a documentação de cada linguagem. O objetivo desta seção é listar os elementos mais úteis para você. Se você tiver alguma dúvida sobre qual a melhor forma de marcar um pedaço específico do seu documento, por favor, envie a sua dúvida para a lista de discussão do projeto de documentação do FreeBSD.

Inline Versus Block

No restante deste documento, quando descrevermos um elemento como *inline* significará que o elemento pode ocorrer dentro de um bloco de elementos, que ele não acarretará em uma quebra de linha. Um elemento *block*, por comparação, irá causar uma quebra de linha (e outros processamentos) quando for encontrado.

4.1. HTML

O HTML, Linguagem de Marcação de Hypertexto, é a linguagem de marcação escolhida para a `World Wide Web`. Maiores informações podem ser encontradas em `http://www.w3.org/`.

O HTML é utilizado para a marcação das páginas do web site do FreeBSD. Ele não deveria ser utilizado (geralmente) para marcar outros tipos de documentos já que o `DocBook` oferece uma maior variedade de elementos. Consequentemente, você só irá encontrar páginas em HTML se estiver escrevendo para o web site.

O HTML já passou por algumas versões, 1, 2, 3.0, 3.2 e a última, 4.0 (disponível nas duas variantes, *strict* e *loose*).

Os HTML DTD's estão disponíveis na coleção de `ports` na pasta textproc/html. Eles são automaticamente instalados como parte do `port` textproc/docproj

4.1.1. Identificador Público Formal (IPF)

Existem vários IPF's para o HTML, os quais dependem da versão (também conhecida como nível) do HTML que você quer declarar compatível com seu documento.

A maioria dos documentos HTML no web site do FreeBSD estão de acordo com a versão `loose` do HTML 4.0.

```
PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
```

4.1.2. Elementos de Seção

Um documento HTML é normalmente dividido em duas partes. A primeira é chamada de *head*, a qual contém meta-informações sobre o documento, tais como o seu título, o nome do autor, o documento pai e assim por diante. A segunda parte, o *body* é contém o conteúdo que vai ser exibido para o usuário.

Estas seções são indicadas pelos elementos `head` e `body`, respectivamente. Esses elementos estão contidos dentro de um elemento `html` de alto-nível.

Example 4.1. Estrutura Normal de um Documento HTML

```
<html>
  <head>
   <title>Título do Documento </title>
  </head>

  <body>

    …

  </body>
</html>
```

4.1.3. Bloco de Elementos

4.1.3.1. Cabeçalho

O HTML permite a denotação de cabeçalho em seu documento, de até seis níveis diferentes.

O maior e mais proeminente cabeçalho é o `h1`, depois vem o `h2`, assim por diante até `h6`.

O conteúdo dos elementos é o texto do cabeçalho.

Example 4.2. `h1`, `h2`, e outras Tags de Header.

Uso:

```
<h1>Primeira seção</h1>

<!-- a introdução do documento entra aqui -->

<h2>Este é o cabeçalho da primeira seção</h2>

<!-- O conteúdo da primeira seção entra aqui -->

<h3>Este é o cabeçalho da primeira subseção</h3>

<!-- O conteúdo da primeira subseção entra aqui -->

<h2>Este é o heading para a segunda seção</h2>

<!-- O conteúdo da segunda seção entra aqui -->
```

Geralmente, uma página HTML deveria ter um cabeçalho de primeiro nível (`h1`). Este poderia conter muitos cabeçalhos de segundo nível (`h2`), os quais por sua vez podem conter muitos cabeçalhos de terceiro nível. Cada elemento h*n* deve ter o mesmo elemento, sendo que os elementos mais acima na hierarquia estarão subtraídos de um. Deve-se evitar buracos na numeração.

Example 4.3. Má organização de elementos `hn`

Uso:

```
<h1>Primeira seção</h1>

<!-- Introdução do documento -->

<h3>Subseção</h3>

<!-- Isto é ruim, não foi usado <h2> -->
```

4.1.3.2. Parágrafos

O HTML suporta elementos formados de um único parágrafo p.

Example 4.4. **p**

Uso:

```
<p>Isto é um parágrafo.  Pode conter qualquer outro elemento</
p>
```

4.1.3.3. Bloco de Citação

Um bloco de citação é uma citação estendida de outro documento que não deveria aparecer no parágrafo atual.

Example 4.5. **blockquote**

Uso:

```
<p>Um pequeno trecho da constituição dos Estados Unidos:</p>

<blockquote>
Nós, povo dos Estados Unidos, com o objetivo de fazer uma
união mais perfeita, estabelecer justiça, garantir
tranquilidade doméstica, promover uma defesa comum, promover
bem estar geral, assegurar as bençõe da liberdade para
nós mesmos e nossa posteridade, organizamos e estabelecemos
essa constituição para os estados Unidos da América.
</blockquote>
```

4.1.3.4. Listas

Você pode apresentar ao usuário três tipos de listas, ordenadas, desordenadas e de definição.

Tipicamente, cada entrada em uma lista ordenada, é numerada enquanto nas listas desordenadas serão processadas por um bullet point. Listas de definição são compostas de duas partes para cada entrada a primeira é o termo a ser definido, e a segunda, é a definição em si.

As Listas ordenadas, desordenadas e de definição, são indicadas pelos elementos `ol`, `ul` e `dl`, respectivamente.

Listas ordenadas e desordenadas contém itens de lista, indicados pelo elemento `li`. Um item de lista pode conter texto, podendo inclusive conter um ou mais elementos `p`.

Listas de definição contém o termo a ser definido (`dt`) e a descrição do termo (`dd`). A definição de um termo só pode conter elementos `inline`. A descrição do termo pode conter elementos do tipo `block`.

Example 4.6. `ul` e `ol`

Uso:

```
<p>Uma lista não ordenada.  Os itens serão provavelmente
precedidos por uma esfera sólida.</p>

<ul>
  <li>Primeiro item</li>

  <li>Segundo item</li>

  <li>Terceiro item</li>
</ul>

<p>Uma lista ordenada, com itens de lista com vários
parágrafos.  Cada item (nota: não cada parágrafo)
será numerado.</p>

<ol>
  <li><p>Este é o primeiro item.  Tem apenas um parágrafo.</
p></li>

  <li><p>Este é o primeiro parágrafo do segundo item.</p>

    <p>Este é o segundo parágrafo do segundo item.</p></li>

  <li><p>Este é o primeiro e único parágrafo do terceiro item.↺
</p></li>
</ol>
```

Example 4.7. Listas de Definição com `dl`

Uso:

```
<dl>
  <dt>Termo 1</dt>

  <dd><p>Parágrafo 1 da definição 1.</p>

    <p>Parágrafo 2 da definição 1.</p></dd>

  <dt>Termo 2</dt>

  <dd><p>Parágrafo 1 da definição 2.</p></dd>

  <dt>Termo 3</dt>

  <dd><p>Parágrafo 1 da definição 3.</p></dd>
</dl>
```

4.1.3.5. Texto Pré-Formatado

Você pode indicar que o texto deve ser apresentado exatamente como esta no arquivo. Tipicamente, isto significa que o texto será mostrado em fonte fixa, múltiplos espaços não serão fundidos em um e que as quebras de linha no texto serão significativas.

Para fazer isto, envolva o conteúdo com o elemento `pre`:

Example 4.8. `pre`

Você pode usar `pre` 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:http://people.FreeBSD.org/~nik/primer/index.html&gt;

  Comments appreciated.

  N
</pre>
```

Tenha em mente que o `<` e o `&` continuam sendo reconhecidos como caracteres especiais em um texto pré-formatado. É por isto que nos exemplos tivemos que

utilizar `<` ao invés de <. Para manter a consistência, o `>` também foi utilizado no lugar do >. Fique atento para caracteres especiais que podem aparecer em textos copiados de origens não formatadas, como por exemplo, de uma mensagem de email ou do código fonte de um programa.

4.1.3.6. Tabelas

Note

A maioria dos navegadores de modo texto (tal como o Lynx) não apresentam tabelas de maneira muito eficiente. Se você quiser que o seu conteúdo seja apresentado em forma de tabelas, você deve considerar outra marcação para evitar problemas.

Marque a informação tabular com o elemento `table`. Uma tabela consiste de uma ou mais linhas (`tr`), cada uma contendo uma ou mais células de dados (`td`). As células podem conter outros elementos de bloco, como parágrafos ou listas. Também pode conter outra tabela (este aninhamento pode ser repetido indefinidamente). Se a célula contém apenas um parágrafo, então não é necessário incluir o elemento `p`.

Example 4.9. Uso Simples de `table`

Uso:

```
<p>Esta é uma simples tabela 2x2.</p>

<table>
  <tr>
    <td>Célula esquerda superior</td>

    <td>Célula direita superior</td>
  </tr>

  <tr>
    <td>Célula esquerda inferior</td>

    <td>Célula direita inferior</td>
  </tr>
</table>
```

Uma célula pode se estender por muitas linhas e colunas. Para indicar isto, coloque o atributo `rowspan` e/ou `colspan`, com valores que indiquem o número de linhas ou colunas a serem ocupados.

Example 4.10. Utiliizando `rowspan`

Uso:

```
<p>Uma célula comprida e estreita na esquerda e duas
células pequenas à direita.</p>

<table>
  <tr>
    <td rowspan="2">Comprida e estreita</td>
  </tr>

  <tr>
    <td>Célula superior</td>

    <td>Célula inferior</td>
  </tr>
</table>
```

Example 4.11. Usando `colspan`

Uso:

```
<p>Uma célula longa em cima, duas células abaixo.</p>

<table>
  <tr>
    <td colspan="2">Célula superior</td>
  </tr>

  <tr>
    <td>Célula inferior esquerda</td>

    <td>Célula inferior direita</td>
  </tr>
</table>
```

Example 4.12. Usando `rowspan` e `colspan` juntos

Uso:

```
<p>Numa tabela 3x3, o bloco superior esquerdo é um conjunto
2x2 fundidos em um.  As outras células são normais.</p>

<table>
  <tr>
    <td colspan="2" rowspan="2">Célula esquerda superior ↺
grande</td>

    <td>Célula direita superior</td>
  </tr>

  <tr>
    <td>Célula do meio a direita</td>
  </tr>

  <tr>
    <td>Célula inferior esquerda</td>

    <td>Célula inferior central</td>

    <td>Célula inferior direita</td>
  </tr>
</table>
```

4.1.4. Elementos In-line

4.1.4.1. Enfatizando a Informação

Você tem dois níveis de ênfase disponíveis em HTML, `em` e `strong`. O `em` é para uma ênfase simples e o `strong` indica uma ênfase mais forte.

Em geral, `em` é apresentada em itálico e `strong` é apresentada em negrito. Mas nem sempre é assim, e você não deve contar com isso.

Example 4.13. `em` e `strong`

Uso:

```
<p><em>Este</em> foi enfatizado
```

```
enquanto <strong>este</strong> foi fortemente enfatizado.</p>
```

4.1.4.2. Negrito e Itálico

Uma vez que o HTML inclui marcação de apresentação, você também pode indicar que um conteúdo deve ser apresentado em negrito ou itálico. Os elementos são `b` e `i` respectivamente.

Example 4.14. `b` e `i`

```
<p><b>Este</b> esta em negrito,
enquanto <i>este</i> está em itálico.</p>
```

4.1.4.3. Indicando Texto de Fonte Fixa

Se você tiver conteúdo que deve ser apresentado em fonte fixa (typewriter), use `tt` (de "teletipo").

Example 4.15. `tt`

Uso:

```
<p> Este documento foi originalmente por
Nik Clayton, que pode ser encontrado por email em
<tt>nik@FreeBSD.org</tt>.</p>
```

4.1.4.4. Tamanho do Conteúdo

Você pode indicar que o conteúdo deve ser apresentado em uma fonte maior ou menor. Existem três maneiras de fazer isto:

1. Use `big` e `small` em torno do texto que você deseja mudar o tamanho. Estas tags podem ser aninhadas, assim `<big><big>Este é muito maior</big></big>` é possível.

2. Use `font` com o atributo `size` ajustado para `+1` ou `-1` respectivamente. Isto tem o mesmo efeito que `big` ou `small`. Entretanto esta forma está ultrapassada.

3. Use `font` com o atributo `size` com um número entre `1` e `7`. O tamanho da fonte padrão é `3`. Esta forma está ultrapassada.

Example 4.16. `big`, `small`, e `font`

Todos os fragmentos fazem a mesma coisa.

```
<p>Este texto é <small>ligeiramente menor</small>.
Mas este texto é <big>ligeiramente maior</big>.</p>

<p>Este texto é <font size="-1">ligeiramente menor</font>.
Mas este texto é <font size="+1">ligeiramente maior</font>.</p>

<p>Este texto é <font size="2">ligeiramente menor</font>.
Mas este texto é <font size="4">ligeiramente maior</font>.</p>
```

4.1.5. Links

Note

Os links também são elementos in-line.

4.1.5.1. Ligando-se a Outros Documentos

Para incluir um link para outro documento na WWW você deve saber o URL do documento ao qual deseja se ligar.

O link é indicado com `a`, e o atributo `href` contém o URL do documento de destino. O conteúdo do elemento se torna o link, e geralmente é indicado para o usuário de alguma maneira (sublinhado, mudança de cor, o formato do cursor do mouse muda quando está sobre ele, etc.

Example 4.17. Usando `<a href="...">`

Uso:

```
<p>Maiores informações estão disponíveis no
```

```
<a href="http://www.FreeBSD.org/">web site do FreeBSD</a>.</p>
```

Este link irá levar o usuário ao topo do documento escolhido.

4.1.5.2. Ligando-se a Outras Partes de um Documento

Para fazer um link a um ponto dentro de outro documento (ou dentro do mesmo documento), é necessário que o autor do documento inclua âncoras ao qual você possa se ligar.

Âncoras são indicadas com `a` e o atributo `name` ao invés de `href`.

Example 4.18. Usando `<a name="...">`

Uso:

```
<p><a name="para1">Este</a> parágrafo pode ser referenciado
em outros links com o nome<tt>para1</tt>.</p>
```

Para fazer um link a uma determinada parte de um documento, faça um link normal para aquele documento, mas inclua o nome da âncora após o símbolo **#**.

Example 4.19. Link para uma determinada parte de outro documento

Suponha que o exemplo `para1` esteja em um documento chamado `foo.html` .

```
<p>Mais informação no <a href="foo.html#para1">primeiro
parágrafo</a> de <tt>foo.html</tt>.</p>
```

Se você for incluir um link para uma âncora dentro do mesmo documento você pode omitir o URL do documento, e usar apenas o nome da âncora (precedido por **#**).

Example 4.20. Links para determinada parte do mesmo documento

Suponha que o exemplo `para1` esteja neste documento

```
<p>Mais informação no <a href="#para1">primeiro
parágrafo</a> deste documento.</p>
```

4.2. DocBook

O DocBook foi originalmente desenvolvido por HaL Computer Systems e O'Reilly & Associates como um DTD para escrever documentação técnica [1]. E desde 1998 tem sido mantido pelo DocBook Technical Committee. Desta forma, ao contrário do LinuxDoc e do HTML, o DocBook é fortemente orientado a marcação que descreve *o que* é alguma coisa, ao invés de descrever *como* ela deve ser apresentada.

Formal Versus Informal

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

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

4.2.1. Extensões FreeBSD

O Projeto de Documentação do FreeBSD ampliou o DocBook DTD adicionando alguns elementos novos. Estes elementos têm como objetivo tornar algumas marcações mais precisas.

Os elementos específicos introduzidos pelo FreeBSD estarão claramente destacados quando forem listados abaixo.

No restante deste documento, o termo “DocBook” deve ser interpretado como sendo a versão do DocBook DTD ampliado pelo projeto de documentação do FreeBSD.

[1] Uma pequena história sobre este assunto pode ser encontrada em http://www.oasis-open.org/docbook/intro.shtml#d0e41.

Note

Não existe nada nestas extenssões que seja específico ao FreeBSD, apenas percebemos que elas seriam melhorias para este projeto em particular. Se alguém dos demais projetos *nix (NetBSD, OpenBSD, Linux, ...) tiver interesse em colaborar para a criação de um conjunto de extensões DocBook padrão, por favor entre em contato com Equipe de Engenharia de Documentação <doceng@FreeBSD.org >.

As extensões FreeBSD não estão (atualmente) na coleção de ports. Elas estão armazenadas na árvore Subversion do FreeBSD, como head/share/xml/freebsd.dtd.

4.2.2. Identificador Formal Público (IFP)

De acordo com as diretrizes DocBook para a escrita de IPF's para adaptação do DocBook, o IPF para o DocBook estendido do FreeBSD é:

```
PUBLIC "-//FreeBSD//DTD DocBook V4.1-Based Extension//EN"
```

4.2.3. Estrutura dos Documentos

DocBook permite que você estruture sua documentação de várias maneiras. No projeto de documentação do FreeBSD nós estamos utilizando dois tipos primários de documentos DocBook: o livro e o artigo.

Um livro é organizado em `chapters` (capítulos). Isso é um requerimento obrigatório. Podem existir `parts` (partes) entre o livro e os capítulos para fornecer mais uma camada de organização. O Handbook, por exemplo, é organizado desta maneira.

Um capítulo pode (ou não) conter uma ou mais sessões. Estas são indicadas pelo elemento `sect1`. Se uma sessão contém outra sessão, então use o elemento `sect2`, e assim por diante até `sect5`.

Capítulos e sessões contém o restante do conteúdo.

Um artigo é mais simples do que um livro, e não usa capítulos. Ao invés disso, o conteúdo de um artigo é organizado em uma ou mais sessões, usando os mesmos elementos `sect1` (e `sect2` e assim por diante) que são usados nos livros.

Obviamente, você deve considerar a natureza da documentação que você esta escrevendo para poder decidir se é melhor uma marcação formatada como um livro ou um artigo. Artigos suportam bem informações que não precisam ser divididas em capítulos, isto é, relativamente pequena, com mais ou menos 20 a 25 páginas de conteúdo. Livros suportam melhor informações que se dividem em capítulos com apêndices e conteúdos similares.

Os tutoriais sobre FreeBSD estão marcados na forma de artigos, enquanto por exemplo este documento, o FAQ do FreeBSD, e o Handbook do FreeBSD estão marcados na forma como livros.

4.2.3.1. Iniciando um Livro

O conteúdo de um livro deve estar contido dentro do elemento `book`. Assim como outros elementos de marcação estrutural, esse elemento pode conter elementos que incluam informações adicionais sobre o livro. Isto é tanto meta-informação, utilizada para fins de referência, ou conteúdo adicional usado para produzir uma página de título.

Estas informações adicionais devem estar contidas dentro do elemento `bookinfo`.

Example 4.21. Modelo padrão `book` com `bookinfo`

```
<book>
  <bookinfo>
    <title>Seu título aqui </title>

    <author>
      <firstname>Seu primeiro nome aqui </firstname>
      <surname>Seu sobrenome </surname>
      <affiliation>
        <address><email>Seu endereço de correio eletrônico ↺
aqui</email></address>
      </affiliation>
    </author>

    <copyright>
      <year>1998</year>
      <holder role="mailto:seu endereço de email ">Seu nome</
holder>
    </copyright>

    <releaseinfo>$FreeBSD$</releaseinfo>

    <abstract>
      <para>Inclua um resumo do conteúdo do seu livro aqui.  </
para>
    </abstract>
  </bookinfo>

  …

</book>
```

4.2.3.2. Começando um Artigo

O conteúdo do artigo deve estar contido dentro do elemento `article`. Assim como outros elementos de marcação estrutural, esse elemento pode conter elementos que incluam informações adicionais sobre o artigo. Isto é tanto meta-informação, utilizada para fins de referência, ou conteúdo adicional usado para produzir uma página de título.

Estas informações adicionais devem estar contidas dentro do elemento `articleinfo`.

Example 4.22. Modelo padrão **`article`** com **`articleinfo`**

```
<article>
  <articleinfo>
    <title>Seu título aqui </title>

    <author>
      <firstname>Seu primeiro nome </firstname>
      <surname>Seu sobrenome </surname>
      <affiliation>
        <address><email>Seu endereço de email </email></address>
      </affiliation>
    </author>

    <copyright>
      <year>1998</year>
      <holder role="mailto:seu endereço de email ">Seu nome</
holder>
    </copyright>

    <releaseinfo>$FreeBSD$</releaseinfo>

    <abstract>
      <para>Inclua um resumo do conteúdo do artigo aqui.  </para>
    </abstract>
  </articleinfo>

  …

</article>
```

4.2.3.3. Indicando Capítulos

Utilize `chapter` para marcar seus capítulos. Cada capítulo tem obrigatoriamente um `title`. Artigos não contêm capítulos, estes são reservados para os livros.

Example 4.23. Um capítulo simples

```
<chapter>
  <title>Título do capítulo</title>

  ...
</chapter>
```

Um capítulo não pode ser vazio; ele precisa conter elementos além do `title`. Se você precisar incluir um capítulo vazio então você precisará incluir um parágrafo vazio.

Example 4.24. Capítulos Vazios

```
<chapter>
  <title>Isto é um capítulo vazio</title>

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

4.2.3.4. Sessões Abaixo dos Capítulos

Em um livro, os capítulos podem (mas não é obrigatoriamente necessário) ser quebrados em seções, subseções, e assim por diante. Em um artigo, as seções são os principais elementos estruturais, e cada artigo deve conter pelo menos uma seção. Utilize o elemento `sect`*n*. O *n* indica o número da seção o qual identifica o nível da seção.

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

Example 4.25. Seções em Capítulos

```
<chapter>
  <title>Um exemplo de capítulo</title>

  <para>Algum texto dentro do capítulo</para>

  <sect1>
    <title>Primeira seção (1.1)</title>
```

```
    &hellip;
  </sect1>

  <sect1>
    <title>Segunda seção (1.2)</title>

    <sect2>
      <title>Primeira subseção (1.2.1)</title>

      <sect3>
        <title>Primeira sub-subseção (1.2.1.1)</title>

        &hellip;
      </sect3>
    </sect2>

    <sect2>
      <title>Segunda subseção (1.2.2)</title>

      &hellip;
    </sect2>
  </sect1>
</chapter>
```

Note

Este exemplo inclui os números das seções nos títulos de seções. Você não deve fazer isso nos seus documentos. A inserção dos números de seção é realizada pelas folhas de estilo (falaremos delas mais a frente), e você não precisa gerenciá-los manualmente.

4.2.3.5. Subdividindo Utilizando o Elemento `Part`

Você pode introduzir outra camada de organização entre `book` e `chapter` utilizando uma ou mais `parts`. Isto não pode ser feito em um `article`.

```
<part>
  <title>Introdução</title>

  <chapter>
    <title>Visão Geral</title>

    ...
  </chapter>

  <chapter>
```

```
    <title>O que é FreeBSD?</title>

    ...
  </chapter>

  <chapter>
    <title>História</title>

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

4.2.4. Elementos de Blocos

4.2.4.1. Parágrafos

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

Na maioria das vezes você só vai precisar usar `para`. O `formalpara` inclui um elemento `title`, e o `simpara` desabilita alguns elementos dentro de `para`. Fique com o `para`.

Example 4.26. `para`

Uso:

```
<para>Isso é um parágrafo.  E pode conter quase
  qualquer outro elemento.</para>
```

Aparência:

Isso é um parágrafo. E pode conter quase qualquer outro elemento.

4.2.4.2. Bloco de Citações

Um bloco de citação é uma longa citação de outro documento que não deveria aparecer no parágrafo atual. Você não irá usá-lo com muita frequência.

Um bloco de citação pode conter opcionalmente um título e uma atribuição (ou eles podem ser deixados sem título e sem atributos)

Example 4.27. `blockquote`

Uso:

```
<para>Um pequeno pedaço da Constituição dos Estados Unidos:</
para>
```

```
<blockquote>
  <title>Preâmbulo da constituição dos Estados Unidos.</title>

  <attribution>Copiado de um site qualquer</attribution>

  <para>Nós, povo dos Estados Unidos, com o objetivo de
  fazer uma união mais perfeita, estabelecer justiça,
  garantir tranquilidade doméstica, promover uma defesa ↺
comum, promover
  bem estar geral, assegurar as bençõs da
  liberdade para nós mesmos e nossa posteridade, organizamos e
  estabelecemos essa constituição para os estados Unidos
  da América.</para>
</blockquote>
```

Aparência:

Um pequeno pedaço da Constituição dos Estados Unidos

> Preâmbulo da constituição dos Estados Unidos.
>
> Nós, povo dos Estados Unidos, com o objetivo de fazer uma união mais perfeita, estabelecer justiça, garantir tranquilidade doméstica, promover uma defesa comum, promover bem estar geral, assegurar as bençõs da liberdade para nós mesmos e nossa posteridade, organizamos e estabelecemos essa constituição para os estados Unidos da América.
>
> —Copiado de um site qualquer

4.2.4.3. Dicas, notas, avisos, informações importantes e sidebars.

Talvez você precise incluir informações extras separadas do corpo principal do texto. Tipicamente isto é "meta" informação que o usuário deve tomar conhecimento.

Dependendo da natureza da informação, devemos utilizar o elemento `tip`, `note`, `warning`, `caution`, ou `important`. Alternativamente, se a informação é relacionada ao corpo principal do texto e não se encaixa em nenhum dos objetos acima, utilize `sidebar`.

As circunstâncias na qual se deve escolher um elemento ao invés do outro não é clara. A documentação do DocBook sugere que:

- A `note` é uma informação que deve ser lida por todos os leitores.
- A `important` é uma variação do `note`.
- A `caution` é usada para informar sobre uma possível perda de dados ou possível dano causado pelo software.

- O `warning` é usado para informações sobre possíveis danos ao hardware, sobre risco de vida ou de ferimento a um membro.

Example 4.28. **`warning`**

Uso:

```
<warning>
<para>Instalar o FreeBSD talvez faça com que você queira
desinstalar o Windows do seu Hard disk.</para>
</warning>
```

Aparência:

Warning

Instalar o FreeBSD talvez faça com que você queira desinstalar o Windows do seu Hard disk.

4.2.4.4. Listas e Procedimentos

Você frequentemente vai precisar listar fragmentos de informação para o usuário, ou apresentar para eles um passo a passo o qual eles devem seguir para alcançar um objetivo específico.

Para fazer isso, utilize `itemizedlist`, `orderedlist`, ou `procedure` [2].

Os elementos `itemizedlist` e `orderedlist` são similares aos seus equivalentes em HTML, `ul` e `ol`. Cada um consiste de um ou mais elementos `listitem`, e cada `listitem` contém um ou mais elementos de bloco. O elemento `listitem` é analogo à `li` do HTML. Entretanto, ao contrário do que ocorre no HTML, aqui elas são obrigatórias.

O elemento `procedure` é ligeiramente diferente. Ele consiste de `steps`, que por sua vez consistem de mais `steps` ou `substeps`. Cada `step` contém elementos de bloco.

Example 4.29. **`itemizedlist`** , **`orderedlist`** , e **`procedure`**

Uso:

[2] Há outros tipos de elementos de lista no DocBook, mas não vamos nos preocupar com eles no momento.

```
<itemizedlist>
  <listitem>
    <para>Esse é o primeiro item enumerado.</para>
  </listitem>

  <listitem>
    <para>Esse é o segundo item enumerado.</para>
  </listitem>
</itemizedlist>

<orderedlist>
  <listitem>
    <para>Esse é o primeiro item ordenado.</para>
  </listitem>

  <listitem>
    <para>Esse é o segundo item ordenado.</para>
  </listitem>
</orderedlist>

<procedure>
  <step>
    <para>Faça isto.</para>
  </step>

  <step>
    <para>Depois faça isto.</para>
  </step>

  <step>
    <para>E agora faça isto.</para>
  </step>
</procedure>
```

Aparência:

- Esse é o primeiro item enumerado.
- Esse é o segundo item enumerado.

1. Esse é o primeiro item ordenado.
2. Esse é o segundo item ordenado.

1. Faça isto.
2. Depois faça isto.
3. E agora faça isto.

4.2.4.5. Mostrando Amostras de Arquivos

Se você quiser mostrar um trecho de um arquivo (ou talvez um arquivo inteiro) ao usuário, use o elemento `programlisting`

Espaços e quebras de linha *são* importantes dentro do `programlisting`. Em particular, isso significa que a tag de abertura deve aparecer na mesma linha que a primeira linha do programa, e a tag de fechamento deve estar na última linha do programa, do contrário linhas em branco espúrias podem ser incluídas.

Example 4.30. `programlisting`

Uso:

```
<para>Quando você tiver terminado, seu programa deve estar ↺
assim:</para>

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

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

Note como os colchetes na linha `#include` precisaram ser referenciados através de suas entidades ao invés de incluídas literalmente.

Aparência:

Quando você tiver terminado, seu programa deve estar assim;

```
#include <stdio.h>

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

4.2.4.6. Chamadas

Uma chamada é um mecanismo para fazer referência a um pedaço de texto ou uma posição específica de um exemplo anterior sem ligar a ele no texto.

Para fazer isso, marque as áreas de interesse no seu exemplo (`programlisting`, `literallayout`, ou o que for) com o elemento `co`. Cada elemento deve ter um `id` único atribuído a ele. Insira um `calloutlist` no final do exemplo acima fazendo referência de volta a ele, exibindo comentários adicionais.

Example 4.31. `co` e `calloutlist`

```
<para>Quando você tivert terminado, seu prograva deve estar ↺
assim;</para>

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

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

<calloutlist>
  <callout arearefs="co-ex-include">
    <para>Inclui o arquivo padrão de IO.</para>
  </callout>

  <callout arearefs="co-ex-return">
    <para>Especifica que <function>main()</function> retorna ↺
um inteiro.</para>
  </callout>

  <callout arearefs="co-ex-printf">
    <para>A chamada <function>printf()</function> que escreve ↺
<literal>hello,
      world</literal> na saída padrão.</para>
  </callout>
</calloutlist>
```

Aparência:

Quando você tiver terminado, seu programa deve estar assim;

```
#include <stdio.h> ❶

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

❶ Inclui o arquivo padrão de IO.

❷ Especifica que `main()` retorna um inteiro.
❸ A chamada `printf()` escreve `hello, world` na saída padrão

4.2.4.7. Tabelas

Ao contrário do HTML, você não precisa usar tabelas para acertar o layout, as folhas de estilo cuidam disto para você. Utilize tabelas apenas para marcação de dados tabulares.

De maneira geral (veja a documentação do DocBook para mais detalhes) uma tabela (que pode ser formal ou informal) consiste de um elemento `table`. O qual contém ao menos um elemento `tgroup`, que especifica (como um atributo) o número de colunas neste grupo da tabela. Dentro do grupo tabela você pode ter um elemento `thead`, que contém os elementos para o cabeçalho da tabela (cabeçalho da coluna), e um `tbody` que contém o corpo da tabela.

Tanto o `tgroup` quanto o `thead` contém elementos `row`, que por sua vez contém elementos `entry`. Cada elemento `entry` especifica uma célula da tabela.

Example 4.32. **`informaltable`**

Uso:

```
<informaltable pgwide="1">
  <tgroup cols="2">
    <thead>
      <row>
        <entry>Este é o cabeçalho da coluna 1</entry>
        <entry>Este é o cabeçalho da coluna 2</entry>
      </row>
    </thead>

    <tbody>
      <row>
        <entry>Linha 1, coluna 1</entry>
        <entry>Linha 1, coluna 2</entry>
      </row>

      <row>
        <entry>Linha 2, coluna 1</entry>
        <entry>Linha 2, coluna 2</entry>
      </row>
    </tbody>
  </tgroup>
</informaltable>
```

Aparência:

Este é o cabeçalho da coluna 1	Este é o cabeçalho da coluna 2
Linha 1, coluna 1	Linha 1, coluna 2
Linha 2, coluna 1	Linha 2, coluna 2

Sempre utilize o atributo `pgwide` com o valor `1` junto ao elemento `informaltable`. Um bug no Internet Explorer pode fazer com que a tabela renderize de forma incorreta se ele for omitido.

Se você não quiser borda na tabela adicione o atributo `frame` ao elemento `informaltable` com o valor `none` (i.e., `<informaltable frame="none">`).

Example 4.33. Tabelas com `frame="none"`

Aparência:

Este é o cabeçalho da coluna 1	Este é o cabeçalho da coluna 2
Linha 1, coluna 1	Linha 1, coluna 2
Linha 2, coluna 1	Linha 2, coluna 2

4.2.4.8. Exemplos para o Usuário Seguir

Muitas vezes você irá precisar mostrar exemplos para que o usuário siga. Normalmente, isso irá consistir de diálogos com o computador, nos quais o usuário digita um comando, e obtém uma resposta de volta, ele digita outro comando e recebe outra reposta, e assim por diante.

Vários elementos e entidades podem ser utilizados nestes casos.

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

  Dentro da marcação do `screen`, espaços em branco são significativos.

`prompt`, `&prompt.root;` e `&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`.

Note

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

`userinput`
: Quanto estiver mostrando um texto que o usuário deva digitar, envolva-o com as tags `userinput`. Ele provavelmente será mostrado diferente para o usuário.

Example 4.34. `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'
```

Note

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 ação do usuário.

4.2.5. Elementos In-line

4.2.5.1. Enfatizando a Informação

Quando você quiser enfatizar uma palavra ou frase em particular, use `emphasis`. Ela pode ser apresentada em itálico, ou negrito, ou pode ser falada diferentemente com um sistema texto-para-fala.

Não há uma maneira de mudar a apresentação da ênfase no seu documento, não existe um equivalente ao `b` e ao `i` do HTML. Se a informação que você está apresentando é importante então considere usar `important` ao invés de `emphasis`.

Example 4.35. `emphasis`

Uso:

```
<para>O FreeBSD é sem dúvida <emphasis>o</emphasis> melhor ↺
sistema operacional tipo Unix
para a arquitetura Intel.</para>
```

Aparência:

O FreeBSD é sem dúvida *o* melhor sistema operacional tipo Unix para a arquitetura Intel.

4.2.5.2. Citações

Para citar um texto de outro documento ou fonte, ou para denotar uma frase que está sendo usada figuradamente, use `quote`. Dentro da tag `quote`, você pode usar a maioria das marcações disponíveis para um texto normal.

Example 4.36. Citações

Uso:

```
<para>Entretanto, certifique-se que a busca não vá além do ↺
<quote>limite entre
  a administração local e pública</quote> como diz o RFC 1535.↺
</para>
```

Aparência:

Entretanto, certifique-se que a busca não vá além do “limite entre a administração local e pública” como diz o RFC 1535.

4.2.5.3. Teclas, Mouse e Combinações

Para se referir a uma tecla específica do teclado, use `keycap`. Para se referir a um botão do mouse, use `mousebutton`. E para se referir a combinação de digitar uma tecla e um clique do mouse, envolva-os com `keycombo`.

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

As folhas de estilo colocam automaticamente o símbolo de ligação, tal como +, entre os nomes das teclas, quando elas estiverem envolvidas com `keycombo`.

Example 4.37. Teclas, Mouse e Combinações

Uso:

```
<para>Para mudar para o segundo terminal virtual, digite
  <keycombo action="simul"><keycap>Alt</keycap>
    <keycap>F1</keycap></keycombo>.</para>

<para>Sair do <command>vi</command> sem salvar o seu ↺
trabalho, digite
  <keycombo action="seq"> <keycap>Esc</keycap><keycap>:</
keycap>
    <keycap>q</keycap><keycap>!</keycap></keycombo>.</para>
```

```
<para>Meu gerenciador de janela é configurado de forma que
  <keycombo action="simul"><keycap>Alt</keycap>
    <mousebutton>botão direito</mousebutton>
  </keycombo> do mouse é usado para mover as janelas.</para>
```

Aparência:

Para mudar para o segundo terminal virtual, digite Alt+F1.

Sair do `vi` sem salvar o seu trabalho, digite Esc : q !.

Meu gerenciador de janela é configurado de forma que Alt+botão direito do mouse é usado para mover as janelas.

4.2.5.4. Aplicações, Comandos, Opções e Citações

Frequentemente você irá fazer referência tanto a aplicações quanto a comandos quando estiver escrevendo a documentação. A distinção entre eles é simples: uma aplicação é o nome de um pacote de programas (ou possivelmente apenas um) que executa uma tarefa em particular. Um comando é o nome de um programa que o usuário pode executar.

Além disso, você eventualmente precisará listar uma ou mais opções que um comando pode aceitar.

Finalmente, você irá querer listar um comando com o número da sua seção no manual, na forma "comando(número)" que é tão comum nos manuais de Unix.

Você deve marcar o nome de uma aplicação com `application`.

Quando você quiser listar um comando com o número da sua seção no manual (o que deve acontecer na maioria das vezes) o elemento do DocBook que deve ser utilizado é o `citerefentry`. Ele irá ter mais dois elementos, `refentrytitle` e `manvolnum`. O conteúdo de `refentrytitle` é o nome do comando, e o conteúdo de `manvolnum` é a seção da página do manual.

Pode ser trabalhoso escrever desta forma, para facilitar este processo foram criadas uma série de entidades gerais. Cada entidade está na forma `&man.manual-page.manual-section;`.

O arquivo que contém estas entidades é o `doc/share/xml/man-refs.ent` , o qual pode ser referenciado utilizando o seguinte FPI:

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

Portanto, a introdução à sua documentação provavelmente será assim:

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

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

…

]>
```

Use o elemento `command` quando quiser incluir um comando "in-line" para ser apresentado como algo que deveria ser digitado pelo usuário.

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

Quando diversas referências a um mesmo comando estiverem próximas é melhor usar a notação `&man.command.section;` para marcar a primeira referência ao mesmo e depois utilizar `command` para marcar as referências subsequentes. Isto fará a saída gerada, especialmente em HTML, ficar visualmente melhor.

Isto pode ser confuso, e muitas vezes a escolha nem sempre é clara. Acredito que o exemplo abaixo ajudará a tornar as coisas mais claras.

Example 4.38. Aplicações, Comandos, Opções

Uso:

```
<para>O <application>Sendmail</application> é a aplicação
  de email mais utilizada em Unix.</para>

<para>O <application>Sendmail</application> inclui os ↺
programas
  <citerefentry>
    <refentrytitle>Sendmail</refentrytitle>
    <manvolnum>8</manvolnum>
  </citerefentry>, &man.mailq.1;, e &man.newaliases.1;.</para>


<para>Um dos parâmetros de linha de comando para o ↺
<citerefentry>
    <refentrytitle>Sendmail</refentrytitle>
    <manvolnum>8</manvolnum>
  </citerefentry>, <option>-bp</option>, ira mostrar o atual ↺
estado da
  mensagem na fila de email.
  Verifique isto na linha de comando usando <command>sendmail -
bp</command>.</para>
```

Aparência:

O Sendmail é a aplicação de email mais utilizada em Unix.

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

Um dos parâmetros de linha de comando para o sendmail(8), `-bp`, irá mostrar o atual estado da mensagem na fila de email. Verifique isto na linha de comando usando `sendmail -bp`.

Note

Veja como a notação `&man.command.section;` é mais fácil de acompanhar.

4.2.5.5. Arquivos, Diretórios, Extensões

Sempre que quiser se referir ao nome de um arquivo, diretório ou uma extensão de arquivo, use o elemento `filename`.

Example 4.39. `filename`

Uso:

```
<para>O fonte SGML do Handbook em Inglês pode ser encotrado em
  <filename class="directory">/usr/doc/en_US.ISO8859-1/books/
handbook/</filename>.
  O primeiro arquivo neste diretório é chamado <filename>book.↺
xml</filename>.
  Você também deve ver o <filename>Makefile</filename>
  e alguns arquivos com a extensão <filename>.ent</filename>.</
para>
```

Aparência:

O fonte SGML do Handbook em Inglês pode ser encotrado em `/usr/doc/en/handbook/`. O primeiro arquivo neste diretório é chamado `handbook.xml` . Você também deve ver o `Makefile` e alguns arquivos com a extensão `.ent`.

4.2.5.6. O Nome dos Ports

Extensões FreeBSD

Estes elementos são parte das extensões do FreeBSD ao DocBook, e não existem no DTD DocBook original.

Você pode precisar incluir o nome de um programa da Coleção de Ports do FreeBSD na documentação. Use a tag `filename` com o atributo `role` ajustado para `package` para identificá-lo. Uma vez que a coleção de ports pode ser instalada em diversos lugares, inclua apenas a categoria e o nome do port, não inclua o `/usr/ports` .

Example 4.40. Tag `filename` com o atributo `package`

Use:

```
<para>Instale o port <filename role="package">net/ethereal</
filename>
para ver o tráfego na rede.</para>
```

Aparência:

Instale o port net/ethereal para ver o tráfego na rede.

4.2.5.7. Dispositivos

Extensões FreeBSD

Estes elementos são parte das extensões do FreeBSD ao DocBook, e não existem no DTD DocBook original.

Você tem 2 opções para se referir a um dispositivo. Você pode se referir ao dispositivo da forma como ele aparece no `/dev`, ou você pode usar o nome do dispositivo como ele aparece no kernel. No segundo caso, use o elemento `devicename`.

Em alguns casos você não terá escolha. Alguns dispositivos, como as placas de rede, não tem entrada no `/dev`, ou as entradas são muito diferentes das esperadas.

Example 4.41. `devicename`

Uso:

```
<para><devicename>sio</devicename> é usado para comunicação
  serial no FreeBSD. <devicename>sio</devicename> se manifesta
  através de entradas em <filename>/dev</filename>, incluindo
  <filename>/dev/ttyd0</filename> e <filename>/dev/cuaa0</
filename>.
  </para>

<para>Por outro lado, dispositivos de rede, tais como ↺
<devicename>ed0</devicename>
  não aparecem <filename>/dev</filename>.
  </para>

<para>No MS-DOS, o primeiro disco flexível é chamado de ↺
<devicename>a:</devicename>.
  No FreeBSD é <filename>/dev/fd0</filename>.
  </para>
```

Aparência:

`sio` é usado para comunicação serial no FreeBSD. `sio` se manifesta através de entradas em `/dev`, incluindo `/dev/ttyd0` e `/dev/cuaa0` .

Por outro lado, dispositivos de rede, tais como `ed0` não aparecem `/dev`.

No MS-DOS, o primeiro disco flexível é chamado de `a:`. No FreeBSD é `/dev/fd0` .

4.2.5.8. Hosts, Domínios, Endereços IP e etc.

Extensões FreeBSD

Estes elementos são parte das extensões do FreeBSD ao DocBook, e não existem no DTD DocBook original.

Você pode marcar a informação sobre a identificação de computadores em rede (hosts) de diversas maneiras. Todas elas usam `hostid` como elemento, com o atributo `role` dizendo o tipo da informação marcada.

Sem o atributo `role`, ou `role="hostname"`
: Sem o atributo `role` (i.e., `hostid` .../`hostid`) a informação marcada é simplesmente o nome do computador, tal como `freefall` ou `wcarchive`. Você pode especificar explicitamente com `role="hostname"` .

`role="domainname"`
: O texto é um nome de domínio, tal como `FreeBSD.org` ou `ngo.org.uk`. Não há o componente do nome do computador (hostname).

`role="fqdn"`
: O texto é um nome completo, com o nome do computador e do domínio. O termo `fqdn` significa "Fully Qualified Domain Name" - Nome de Domínio Completamente Qualificado.

`role="ipaddr"`
: O texto é um endereço IP, provavelmente expresso como `dotted quad`.

`role="ip6addr"`
: O texto é um endereço IPv6.

`role="netmask"`
: O texto é uma máscara de rede, que pode ser expressa como `dotted quad`, uma string hexadecimal ou como `/` seguido de um número.

`role="mac"`
: O texto é um endereço MAC Ethernet, expresso como números hexadecimais de 2 digitos separados por dois pontos (:).

Example 4.42. `Hostid` e `Roles`

Use:

```
<para>A máquina local sempre pode ser referida pelo nome
<hostid>localhost</hostid>, que terá o endereço IP
<hostid role="ipaddr">127.0.0.1</hostid>.</para>


<para>O domínio <hostid role="domainname">FreeBSD.org</hostid>
contém vários hosts diferentes, incluindo
<hostid role="fqdn">freefall.FreeBSD.org</hostid> e
<hostid role="fqdn">pointyhat.FreeBSD.org</hostid>.</para>

<para>Quando estiver adicionando um apelido para uma ↺
interface (usando
<command>ifconfig</command>) <emphasis>sempre</emphasis> use a
máscara de rede <hostid role="netmask">255.255.255.255</
hostid> (que
```

```
também pode ser expressa como <hostid ↺
role="netmask">0xffffffff</hostid>).
</para>

<para>O endereço MAC identifica unicamente cada placa de rede
existente.  Um endereço MAC típico se parece com  <hostid
role="mac">08:00:20:87:ef:d0</hostid>).</para>
```

Aparência:

A máquina local sempre pode ser referida 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`.

Quando estiver adicionando um apelido para uma interface (usando `ifconfig`) *sempre* use a máscara de rede `255.255.255.255` (que também pode ser expressa como `0xffffffff`).

O endereço MAC identifica unicamente cada placa de rede existente. Um endereço MAC típico se parece com `08:00:20:87:ef:d0`.

4.2.5.9. Usernames

Extensões FreeBSD

Estes elementos são parte das extensões do FreeBSD ao DocBook, e não existem no DTD DocBook original.

Quando precisar se referir a um nome de usuário específico, tal como `root` ou `bin`, use o elemento `username`.

Example 4.43. `Username`

Uso:

```
<para>Para executar a maioria das funções administrativas ↺
você precisará
  ser <username>root</username>.</para>
```

Aparência:

Para executar a maioria das funções administrativas você precisará ser `root`.

4.2.5.10. Descrevendo `Makefiles`

Extensões FreeBSD

Estes elementos são parte das extensões do FreeBSD ao DocBook, e não existem no DTD DocBook original.

Existem dois elementos para descrever partes de `Makefiles`, `maketarget` e `makevar`.

O `maketarget` identifica uma alvo de construção exportado por um `Makefile` que pode ser passado como um parâmetro ao `make`. O `makevar` identifica uma variável que pode ser ajustada (no ambiente, na linha de comando do `make`, ou dentro do `Makefile`) para influenciar o processo.

Example 4.44. `Maketarget` e `Makevar`

Uso:

```
<para>Dois alvos comuns em um <filename>Makefile</filename> são
<maketarget>all</maketarget> e <maketarget>clean</maketarget>.↺
</para>

<para>Tipicamente, usar <maketarget>all</maketarget> irá ↺
refazer a
aplicação, usar <maketarget>clean</maketarget> irá remover os
arquivos temporários (<filename>.o</filename> por exemplo) ↺
criados
pelo processo de construção.</para>

<para><maketarget>clean</maketarget> pode ser controlado por
muitas variáveis, incluindo <makevar>CLOBBER</makevar> e
<makevar>RECURSE</makevar>.</para>
```

Aparência:

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

Tipicamente, usar `all` irá refazer a aplicação, usar `clean` irá remover os arquivos temporários (`.o` por exemplo) criados pelo processo de construção.

O `clean` pode ser controlado por muitas variáveis, incluindo `CLOBBER` e `RECURSE`.

4.2.5.11. Texto Literal

Com frequência você irá precisar incluir texto "literal" na documentação. Este texto que é originado de outro arquivo, ou que deve ser copiado de forma fiel da documentação para outro arquivo.

Às vezes, o `programlisting` será suficiente. Mas o `programlisting` nem sempre é apropriado, particularmente quando você quer incluir uma parte de um arquivo "in-line" com todos os parágrafos.

Nestas ocasiões, use `literal`.

Example 4.45. `Literal`

Uso:

```
 <para>A linha <literal>maxusers 10</literal> no arquivo de
configuração do kernel determina o tamanho de muitas tabelas
do sistema, e diz aproximadamente quantos logins simultâneos
o sistema irá suportar.</para>
```

Aparência:

A linha `maxusers 10` no arquivo de configuração do kernel determina o tamanho de muitas tabelas do sistema, e diz aproximadamente quantos logins simultâneos o sistema irá suportar.

4.2.5.12. Mostrando Itens que o Usuário Deve Preencher

Muitas vezes você irá querer mostrar ao usuário o que fazer, ou precisará se referir a um arquivo, a uma linha de comando ou coisa parecida, na qual ele não poderá simplesmente copiar o exemplo que você forneceu, mas na qual ele terá deve dar alguma informação por ele mesmo.

O elemento `replaceable` foi criado para estas ocasiões. Use ele dentro de outros elementos para indicar partes do conteúdo do elemento que o usuário deve substituir.

Example 4.46. `replaceable`

Uso:

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

Aparência:

```
% man comando
```

O `replaceable` pode ser usado em muitos elementos diferentes, incluindo `literal`. Este exemplo também mostra que `replaceable` só deve envolver o conteúdo que o usuário *deve* fornecer. O outro conteúdo não deve ser alterado.

Uso:

```
<para>A linha <literal>maxusers <replaceable>n</replaceable></
literal>
no arquivo de configuração do kernel determina o tamanho de ↺
muitas
tabelas do sistema, e diz aproximadamente quantos logins ↺
simultâneos
o sistema irá suportar.
</para>

<para>Para uma estação de trabalho, <literal>32</literal> é um
bom valor para <replaceable>n</replaceable>.</para>
```

Aparência:

A linha `maxusers n` no arquivo de configuração do kernel determina o tamanho de muitas tabelas do sistema, e diz aproximadamente quantos logins simultâneos o sistema irá suportar.

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

4.2.5.13. Citando erros do sistema

Você pode querer mostrar erros gerados pelo FreeBSD. Marque-os com `errorname`. Isto indicará o erro exato que aparece.

Example 4.47. `errorname`

Uso:

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

Aparência:

```
Panic: cannot mount root
```

4.2.6. Imagens

Important

O suporte a imagem na documentação ainda é extremamente experimental. O mecanismo aqui descrito provavelmente não irá mudar, mas isto não é garantido.

Você precisará instalar o port graphics/ImageMagick que é usado para converter entre os diferentes formatos de imagens. Ele é grande e a sua maior parte não é necessária. Entretanto, enquanto nós ainda estamos trabalhando nos `Makefiles` e em outros itens da infraestrutura, ele torna as coisas mais fáceis. Este port *não* está no meta port textproc/docproj, você deve instalá-lo manualmente.

O melhor exemplo do que acontece na prática é o documento `doc/en_US.ISO8859-1/articles/vm-design/` . Se você se sentir inseguro na descrição que segue, dê uma olhada nos arquivos deste diretório e veja como tudo se encaixa. Experimente criar versões com diferentes formatos de saída do documento para ver como a marcação de imagem aparece no documento final.

4.2.6.1. Formatos de Imagens

Atualmente suportamos dois formatos de imagens. O formato que você deve usar depende da natureza da sua imagem.

Para imagens vetoriais, tais como diagramas de rede, linhas temporais e similares, use Encapsulated Postscript, e certifique-se que suas imagens tenham a extensão `.eps`.

Para bitmaps, tais como a captura de uma tela, use o formato Portable Network Graphic, e certifique-se que suas imagens tenham a extensão `.png`.

Estes são os *únicos* formatos de imagem que podem ser gravados no repositório Subversion.

Use o formato correto para a imagem correta. Espera-se que a sua documentação tenha tanto imagens EPS quanto PNG. Os `Makefiles` asseguram que o formato correto seja es-

colhido dependendo do formato de saída da sua documentação. *Não faça commit da mesma imagem nos dois formatos diferentes para o repositório.*

Important

É esperado que o projeto de documentação passe a utilizar no futuro o formato Scalable Vector Graphic (SVG) para imagens vetoriais. Entretanto, o atual estado das ferramentas de edição torna isto impraticável.

4.2.6.2. Marcação

A marcação para uma imagem é relativamente simples. Primeiro, marque um `mediaobject`. O `mediaobject` pode conter outros objetos mais específicos. Estamos interessandos em dois, o `imageobject` e o `textobject`.

Você deve incluir um `imageobject`, e dois elementos `textobject`. O `imageobject` irá apontar para o nome do arquivo da imagem que será usada (sem a extensão). Os elementos `textobject` contém a informação que será apresentada ao usuário junto, ou não, com a imagem.

Há duas circunstâncias em que isso pode ocorrer.

- Quando o leitor estiver vendo a documentação em HTML. Neste caso, cada imagem precisará ter um texto alternativo associado para mostrar ao usuário, tipicamente enquanto a imagem estiver sendo carregada, ou se ele parar o ponteiro do mouse sobre a imagem.
- Quando o leitor estiver vendo a documentação em modo texto. Neste caso, cada imagem deve ter uma ASCII art equivalente para mostrar ao usuário.

Um exemplo provavelmente irá tornar as coisas mais fáceis de se entender. Suponha que você tenha uma imagem, chamada `fig1.png` , que você queira incluir no documento. Esta imagem é um retângulo com um A dentro dele. A marcação para isso será:

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

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

```
  <textobject>
    <phrase>Uma figura</phrase> ❸
  </textobject>
</mediaobject>
```

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

❷ O primeiro `textobject` deve conter um elemento `literallayout`, onde o atributo `class` é ajustado para `monospaced`. Esta é a oportunidade para você demonstrar suas habilidades com ASCII art. O conteúdo será usado se o documento for convertido para texto puro.

Veja como as primeras e últimas linhas do conteúdo do elemento `literallayout` fica próximo a tag do elemento. Isso assegura que não sejam incluídos espaços extras.

❸ O segundo `textobject` deve conter um único elemento `phrase`. O conteúdo deste elemento se tornará o atributo `alt` para as imagens quando o documento for convertido para HTML.

4.2.6.3. Entradas no `Makefile`

Suas imagens devem estar listadas na variável `IMAGES` do `Makefile`. Esta variável deve conter o nome de todos *fontes* das suas imagens. Por exemplo, se você criou três figuras, `fig1.eps` , `fig2.png` , `fig3.png` , então o seu `Makefile` deve ter linhas como estas.

```
…
IMAGES= fig1.eps fig2.png fig3.png
…
```

ou

```
…
IMAGES=  fig1.eps
IMAGES+= fig2.png
IMAGES+= fig3.png
…
```

De novo, o `Makefile` irá fazer uma lista completa das imagens necessárias para criar o seu documento fonte, você precisa listar apenas as imagens que *você* fornece.

4.2.6.4. Imagens e Capítulos em Subdiretórios

Você precisa ser cuidadoso quando separar a sua documentação em arquivos menores (veja Section 3.7.1, “Utilizando entidades gerais para incluir arquivos”) em diretórios diferentes.

Suponha que você tenha um livro com três capítulos, e os capítulos estão armazenados cada um no seu próprio diretório, chamados `chapter1/chapter.xml` , `chap-`

`ter2/chapter.xml` , e `chapter3/chapter.xml` . Se cada capítulo tiver imagens associadas a eles, é sugerido que você as coloque dentro do respectivo subdiretório de cada capítulo (`chapter1/`, `chapter2/`, `chapter3/`).

Entretanto, se você fizer isso você deve incluir o nome dos diretórios na variável `IMAGES` no `Makefile`, *e* deve incluir o nome do diretório no elemento `imagedata` do seu documento.

Por exemplo, se você 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
…
```

Desta forma tudo deve funcionar.

4.2.7. Links

Note

Links também são elementos in-line.

4.2.7.1. Ligando-se a outras partes no mesmo documento

Links dentro do mesmo documento exigem que você especifique de onde você esta se ligando (i.e., o texto no qual o usuário irá clicar, ou indicando de outra maneira a origem do link) e para onde você está se ligando (o destino do link).

Cada elemento dentro do DocBook tem um atributo chamado `id`. Você pode por um texto neste atributo para identificar unicamente o elemento associado a ele.

Este valor será usado quando você especificar a origem do link.

Normalmente, você irá se ligar apenas a capítulos ou seções, assim você irá colocar o atributo `id` nestes elementos.

Example 4.48. O atributo `id` em capítulos ou seções

```
<chapter id="chapter1">
  <title>Introdução</title>

  <para>Esta é a introdução.  Contém uma
    subseção que também será identificada.
  </para>
  <sect1 id="chapter1-sect1">
    <title>Subseção 1</title>

    <para>Esta é a subseção.</para>
  </sect1>
</chapter>
```

Obviamente, você deve usar nomes mais descritivos. Estes nomes devem ser únicos dentro do documento (i.e., não apenas no arquivo, mas sim no documento no qual o arquivo está incluído). Repare como o `id` é construído adicionando-se texto ao `id` do capítulo. Isto ajuda a garantir que eles sejam únicos.

Se você quiser permitir ao usuário saltar para uma parte específica do documento (possivelmente para o meio do parágrafo ou um exemplo), use o elemento `anchor`. Este elemento não tem conteúdo, mas aceita o atributo `id`.

Example 4.49. `anchor`

```
<para>
Este parágrafo tem um <anchor id="para1">alvo dentro dele.
O qual não irá aparecer no documento
</para>
```

Quando você quiser dar ao usuário um link que possa ser ativado (provavelmente clicando-se) para ir para outra seção do documento que tenha um atributo `id`, você pode usar `xref` ou `link`.

Ambos os elementos têm um atributo `linkend`. O valor deste atributo deve ser o mesmo usado no atributo `id` (não importa se ele ainda não ocorreu no documento; isto funciona tanto para um link à frente quanto para trás).

Se você utilizar o `xref` então você não terá controle sobre o texto do link. Ele será gerado para você.

Example 4.50. Usando `xref`

Assuma que este fragmento apareça em algum lugar de um documento que inclua o `id` do exemplo.

```
<para>Maiores informações podem ser encontradas
  em <xref linkend="chapter1"/></para>

<para>Informações mais específicas podem ser
  encontradas na <xref linkend="chapter1-sect1"/>.</para>
```

O texto do link será gerado automaticamente, e irá se parecer com (As palavras em *itálico* indicam o texto que será o link):

> Maiores informações podem ser encontradas no *Capítulo Um*.
>
> Informações mais específicas podem ser encontradas na *seção chamada Subseção 1*.

Observe como o texto do link é deduzido a partir do título da seção ou do número do capítulo.

Note

Isto significa que você *não pode* usar `xref` para se ligar a um atributo `id` em um elemento `anchor`. O `anchor` não tem conteúdo, desta forma o `xref` não pode gerar o texto para o link.

Se você quiser controlar o texto do link você deverá utilizar `link`. Este elemento envolve o conteúdo, e o conteúdo será usado para o link.

Example 4.51. Usando `link`

Assuma que este fragmento aparece em algum lugar em um documento que inclui o `id` do exemplo.

```
<para>Maiores informações podem ser encontradas
<link linkend="chapter1">no primeiro capítulo</link>.
</para>

<para>Informações mais específicas podem ser encontradas
<link linkend="chapter1-sect1">nesta</link> seção
</para>
```

Isto irá gerar o seguinte (Palavras em *itálico* indicam o texto que será o link):

> Maiores informações podem ser encontradas *no primeiro capítulo* .
>
> Informações mais específicas podem ser encontradas *nesta* seção

Note

Este último é um mau exemplo. Nunca use palavras como “esta” ou “aqui” como origem do link. O leitor terá que procurar no contexto próximo para ver para onde o link o está levando.

Note

Você *pode* usar o elemento `link` para incluir um link para um `id` em um elemento `anchor`, uma vez que o conteúdo de `link` define o texto que será usado para o link.

4.2.7.2. Ligando-se a documentos na WWW

Ligar-se a um documento externo é muito mais simples, desde que você saiba o URL do documento ao qual deseja se ligar. Basta utilizar o elemento `ulink`. O atributo `url` é o URL da página para o qual o link aponta, e o conteúdo do elemento é o texto que será mostrado para o usuário ativar.

Example 4.52. `ulink`

Uso:

```
<para>É claro que você pode parar de ler este documento e ir ↺
para a
<ulink url="&url.base;/index.html">Página principal do ↺
FreeBSD</ulink>
</para>
```

Aparência:

É claro que você pode parar de ler este documento e ir para a Página principal do FreeBSD

Chapter 5. * Folhas de Estilo

O SGML não diz nada sobre como um documento deve ser exibido ao usuário, ou formatado para impressão. Para fazer isto, várias linguagens foram desenvolvidas para descrever folhas de estilo, incluindo DynaText, Panorama, SPICE, JSSS, FOSI, CSS, e DSSSL.

Para o `DocBook`, nós estamos usando folhas de estilo escritas em DSSSL. Para o HTML nós estamos usando o CSS.

5.1. * DSSSL

O projeto de documentação usa uma versão ligeiramente customizada das folhas de estilo modulares do `DocBook` de Norm Walsh.

Elas podem ser encontradas no textproc/dsssl-docbook-modular.

As folhas de estilo modificadas não estão no sistema de `ports`. Elas são parte do repositório de fontes do projeto de documentação, e podem ser encontradas em `doc/share/xml/freebsd.dsl` . Elas estão bem comentadas, e como a conclusão desta seção está pendente, recomendamos que você examine este arquivo para ver como algumas das opções disponíveis nas folhas de estilo padrão foram configuradas para customizar a saída para o projeto de documentação do FreeBSD. Este arquivo também contém exemplos que mostram como estender os elementos que a folha de estilo compreende, que é como os elementos específicos para o FreeBSD foram formatados.

5.2. CSS

Folha de estilo em cascata (CSS) é um mecanismo para anexar a informação de estilo (fontes, peso, tamanho, cor, e assim por diante) aos elementos de um documento HTML sem abusar do HTML para fazê-lo.

5.2.1. Documentos DocBook

As folhas de estilo DSSSL do FreeBSD incluem uma referência para a folha de estilo, `docbook.css` , a qual espera-se aparecer no mesmo diretório dos arquivos HTML. Este arquivo CSS geral do projeto é copiado de `doc/share/misc/docbook.css` quando os documentos são convertidos para HTML, e é instalado automaticamente.

Chapter 6. Estruturando Documentos Sob doc/

A árvore doc/ é organizada de uma forma particular, e os documentos que compõe o *Primer* do Projeto de Documentação do FreeBSD devem ser por isso organizados de forma particular. O objetivo é tornar simples a adição de nova documentação à árvore, e:

1. Facilitar a automatização da conversão de documentos para outros formatos.

2. Promover consistência entre diferentes formas de organizar a documentação, facilitar a troca entre diferentes documentos.

3. Facilitar a decisão de onde na árvore uma nova documentação deveria ser colocada.

Além disso, a árvore de documentação tem de acomodar documentação que pode estar em muitas diferentes línguas e muitas diferentes codificações. É importante que a estrutura da árvore de documentação não force nenhum padrão particular ou preferência cultural.

6.1. O Nível Superior, doc/

Existem dois tipos de diretórios sob doc/, cada um com nomes de diretórios e significados muito específicos.
Diretório: share/
Significado: Contém arquivos que não são específicos as várias traduções e codificações da documentação. Contém subdiretórios para promover uma melhor categorização da informação. Por exemplo, os arquivos que compõem a infraestrutura de make(1) encontram-se em share/mk, enquanto os arquivos adicionais para suporte SGML (como as extensões do FreeBSD ao DocBook DTD) encontram-se em share/xml.
Diretório: idioma.codificação/
Significado: Existe um diretório para cada tradução e codificação da documentação, por exemplo en_US.ISO8859-1/ e zh_TW.Big5/. Os nomes são longos, mas ao especificar completamente a língua e codificação prevenimos qualquer futura dor de cabeça caso um time de tradução queira prover a documentação na mesma língua mas em mais de uma codificação. Isto também nos isola completamente de quaisquer problemas que possam ser causados por uma mudança para Unicode.

6.2. Os Diretórios de idioma.codificação

Estes diretórios contêm os documentos propriamente ditos. A documentação é dividida em até três categorias adicionais neste nível, indicadas pelos diferentes nomes de diretórios.

Diretório: `articles`
Conteúdo: Documentação codificada como DocBook `article` (ou equivalente). Razoavelmente pequena, e separada em seções. Normalmente disponível apenas como um arquivo HTML.
Diretório: `books`
Conteúdo: Documentação codificada como DocBook `book` (ou equivalente). Com o tamanho de um livro, e separada em capítulos. Normalmente disponível tanto como um grande arquivo HTML (para pessoas com conexões rápidas, ou que queiram imprimí-la facilmente a partir de um navegador Internet) quanto como uma coleção de pequenos arquivos interligados.
Diretório: `man`
Conteúdo: Para traduções das páginas de manual do sistema. Este diretório conterá um ou mais diretórios `mann`, correspondendo as seções que foram traduzidas.

Nem todo diretório `idioma.codificação` conterá todos estes diretórios. Isto depende de quantos documentos já foram traduzidos pelo time de tradução.

6.3. Informação Específica do Documento

Esta sessão contém observações específicas sobre documentos particulares controlados pelo FDP.

6.3.1. O Handbook

`books/handbook/`

O Handbook é escrito de forma a obedecer a versão estendida do DTD DocBook utilizado pelo projeto FreeBSD.

O Handbook é organizado como um `book` Docbook. Ele está dividido em `part`es, e cada uma delas pode conter diversos `chapter`s (Capítulos). Os `chapter`s estão subdivididos em seções (`sect1`) e subseções (`sect2`, `sect3`) e assim por diante.

6.3.1.1. Organização Fisíca

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

Note

A organização do Handbook pode mudar ao longo do tempo, e este documento pode ficar defasado no detalhamento destas alterações organizacionais. Se você tiver alguma pergunta sobre como o Hand-

> book é organizado, por favor entre em contato com a lista de discussão do projeto de documentação do FreeBSD.

6.3.1.1.1. Makefile

O Makefile define algumas variáveis as quais afetam a forma como o fonte SGML é convertido para outros formatos, e lista os vários arquivos fonte que compõem o Handbook. Ele também inclui um arquivo padrão chamado doc.project.mk , o qual contém o restante do código responsável por realizar a conversão dos documentos de um formato para outro.

6.3.1.1.2. book.xml

Este é o documento de mais alto nível do Handbook. Ele contém as declarações DOCTYPE do Handbook, assim como os elementos que descrevem a estrutura do Handbook.

O book.xml utiliza entidades de parâmetro para carregar os arquivos com extensão .ent. Estes arquivos (descritos abaixo) definem as entidades gerais as quais são utilizadas ao longo de todo o Handbook.

6.3.1.1.3. directory/chapter.xml

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

Por exemplo, se um dos arquivos de capítulos contiver:

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

Então ele será chamado de chapter.xml e será armazenadao no diretório kernelconfig . Em geral, todo o conteúdo do capítulo será mantido neste arquivo.

Quando a versão HTML do Handbook for produzida, será gerado um arquivo kernelconfig.html . Isto ocorre devido ao valor do atributo id e não está relacionado ao nome do diretório.

Nas versões anteriores do Handbook os arquivos eram armazenados no mesmo diretório que o book.xml , e depois nomeados a partir do valor do atributo id presente no elemento chapter do arquivo. Agora, é possível incluir imagens em cada capítulo. As imagens de cada capítulo do Handbook são armazenadas dentro de share/images/books/handbook. Observe que as versões localizadas destas imagens devem ser colocadas no mesmo diretório com o código fonte SGML de cada capítulo. Colisões de *namespace* são inevitáveis,

e é muito mais simples trabalhar com vários diretórios que contenham poucos arquivos em cada um, do que trabalhar com um diretório que contenha muitos arquivos.

Um exame rápido vai mostrar que existem muitos diretórios com um único arquivo `chapter.xml` , incluindo `basics/chapter.xml` , `introduction/chapter.xml` , e `printing/chapter.xml` .

Important

Os capítulos e/ou diretórios não devem ser nomeados de forma que reflitam sua ordem no Handbook. Esta ordenação pode mudar com uma reorganização do conteúdo do Handbook; este tipo de reorganização não deve (geralmente) incluir a necessidade de renomear os arquivos (a menos que um capítulo inteiro esteja sendo promovido ou rebaixado na hierarquia).

Cada arquivo `chapter.xml` não será um documento SGML completo. Em particular, eles não terão as suas próprias linhas DOCTYPE no início do arquivo.

Isto é uma infelicidade pois torna impossível tratá-los como arquivos SGML genéricos e simplesmente convertê-los para HTML, RTF, PS, e outros formatos da mesma forma que o Handbook principal é gerado. Isto irá forçá-lo a reconstruir o Handbook inteiro sempre que você desejar ver o efeito de uma alteração realizada em apenas um capítulo.

Chapter 7. O processo de construção da documentação

A principal finalidade desse capítulo é explicar claramente *como o processo de criação da documentação é organizado*, e *como fazer modificações a este processo*.

Depois de finalizar a leitura deste capítulo você deverá:

- Saber o que você precisa para compilar a documentação mantida pelo FDP, em adição ao que foi mencionado no capítulo Ferramentas SGML.
- Ser capaz de ler e entender as instruções do make que estão presentes em cada documento `Makefile`, assim como ter uma visão geral do `doc.project.mk` .
- Ser capaz de customizar o processo de compilação usando variáveis e alvos do make.

7.1. Ferramentas para construção da documentação do FreeBSD

Aqui estão suas ferramentas. Use-as de todas as formas que puder.

- A primeira ferramenta que você precisará é o make, mais especificamente o Berkeley Make.
- A construção de pacotes no FreeBSD é executada pelo pkg_create. Se você não está utilizando o FreeBSD, você terá que viver sem o uso de pacotes, ou então terá que compilar o código fonte você mesmo.
- O gzip é necessário para criar versões compactadas do documento. O compressor bzip2 e os arquivos zip também são suportados. O tar é suportado, e a construção de pacotes necessita dele.
- O install é o método padrão para instalar a documentação. Entretanto, existem alternativas.

Note

É improvável que você tenha qualquer problema em localizar esses dois últimos, eles estão sendo mencionados apenas para que a listagem fique completa.

7.2. Entendendo Makefiles na árvore da documentação

Há três tipos principais de Makefiles na árvore do projeto de documentção do FreeBSD.

- Os Makefiles de subdiretório simplesmente passam comandos para os diretórios abaixo dele.
- Os Makefiles de documentação descrevem o(s) documento(s) que deve(m) ser produzido(s) a partir deste diretório.
- Os Make includes são os responsáveis pela produção do documento, e geralmente possuem o nome no formato doc.xxx.mk .

7.2.1. Makefiles de Subdiretórios

Estes Makefiles geralmente tem a forma:

```
SUBDIR =articles
SUBDIR+=books

COMPAT_SYMLINK = en

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

Resumidamente, as primeiras quatro linhas não vazias definem as variáveis do make, SUBDIR, COMPAT_SYMLINK , e DOC_PREFIX .

A primeira declaração da variável SUBDIR, tanto quanto a declaração da variável COMPAT_SYMLINK , mostra como atribuir um valor a uma variável, sobrescrevendo qualquer valor anterior que a mesma contenha.

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

A declaração do DOC_PREFIX mostra como um valor é atribuído para uma variável, mas somente se ela ainda não estiver definida. Isto é útil se o 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? O 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 (por exemplo o 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 que contém o caminho para o diretório atual.

A linha final inclui o arquivo principal do projeto de documentação do FreeBSD, o doc.project.mk , ele é o responsável por converter estas variáveis em instruções de compilação para uso do make.

7.2.2. Makefiles de Documentação

Estes Makefiles ajustam várias variáveis do make as quais descrevem como construir a documentação contida em um determinado 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 é uma muito importante. Esta variável fornece a habilidade de reivindicar a propriedade sobre um documento no projeto de documentação do FreeBSD, é por meio dela que você recebe a responsabilidade de mantê-lo.

DOC é o nome (sem a extensão .xml) do principal documento criado por este diretório. A variável 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.

O `FORMATS` indica os formatos nos quais o documento deve ser gerado por padrão. O `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.

Note

Nós abordamos a atribuição das variáveis opcionais na seção anterior.

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

7.3. Includes do Make do projeto de documentação do FreeBSD

Isto é melhor explicado pela inspeção no código. Aqui estão os arquivos include do sistema:

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

7.3.1. `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"
```

7.3.1.1. 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.

O `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 (US English) é o padrão.

Note

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.

7.3.1.2. Condicionais

A linha `.if defined(DOC)` é um exemplo da condicional do make , 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.

7.3.2. doc.subdir.mk

Este arquivo é muito longo para ser explicado por inspeção, você deve ser capaz de interpretá-lo com o conhecimento adquirido nos capítulos anteriores, e com a pequena ajuda dada aqui.

7.3.2.1. 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`).
- O `COMPAT_SYMLINK` já foi descrito na seção Makefiles de subdiretório .

7.3.2.2. Targets e Macros

As dependências são descritas por `target: dependência1 dependência2 ...` , nas quais, para construir o `target`, você necessita 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 define esta macro a parte 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` irá usar a macro `_SUBDIRUSE` depois de ter executado a instrução `rm -f ${CLEANFILES}`. De fato, isto causa uma limpeza (`clean`) na árvore de diretórios, deletando os arquivos construídos enquanto vai *descendo* pelos subdiretórios, e não quando vai na direção oposta.

7.3.2.2.1. Targets fornecidos

- `install` e `package`, ambos descem pela árvore de diretórios executando a sua versão real dentro dos subdiretórios. (`realinstall` e `realpackage` respectivamente).

- O `clean` remove os arquivos criados pelo processo de compilação (e também desce na árvore de diretórios). O `cleandir` faz a mesma coisa, e também remove o diretório de objetos se este existir.

7.3.2.3. 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.

7.3.2.4. Construção de Looping no `make (.for)`

O `.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 o `.for` e o `.endfor` serão repetidas para cada elemento, com o `entry` sendo substituído com o valor do elemento atual.

Chapter 8. O Website

8.1. Preparação

Utilize um disco que tenha espaço livre suficiente. Você irá precisar de 200 MB a 500 MB, dependendo do método que escolher. Este espaço irá abrigar as ferramentas SGML, um subconjunto da árvore svn, os arquivos temporários de trabalho e as páginas web instaladas.

Note

Certifique-se que seus ports de documentação estão atualizados! Quando na dúvida, remova os ports antigos usando o comando pkg_delete(1) antes de instalar o port. Por exemplo, nós atualmente dependemos do jade-1.2, e se você tem instalado o jade-1.1, por favor execute:

```
# pkg_delete jade-1.1
```

8.1.1. Usando o `svn`

O `svn` é necessário para se obter ("`check out`") os arquivos do `doc/` a partir do repositório Subversion. O `svn` pode ser instalado com o pkg_add(1) ou a partir da coleção de Ports do FreeBSD, executando:

```
# cd /usr/ports/devel/subversion
# make install clean
```

Para obter os arquivos com o código fonte completo do web site do FreeBSD, execute:

```
# svn checkout svn://svn.FreeBSD.org/doc/head/ /usr/build
```

Tip

Se o comando `svn` não estiver sendo executado pelo usuário `root`, certifique-se de que o diretório `/usr/build` possui as permissões adequadas ao usuário utilizado. Se não for possível alterar as per-

> missões, utilize um diretório diferente para armazenar os arquivos do web site.

Quando o `svn` terminar, a versão atual do website do FreeBSD estará disponível em `/usr/build`. Se você utilizou um diretório alvo diferente, substitua o `/usr/build` apropriadamente ao longo do restante deste documento.

É isso! Agora você pode proceder com a geração do web site.

8.2. Construa as páginas web do início

Depois de ter completado os passos necessários para obter os arquivos com o código fonte do website, você estará pronto para iniciar a compilação do web site. No nosso exemplo, o diretório de compilação é `/usr/build` e todos os arquivos necessários já estão disponíveis no mesmo.

1. Vá para o diretório de compilação:

```
# cd /usr/build
```

2. A compilação do web site deve ser iniciada de dentro do diretório `en_US.ISO8859-1/htdocs` executando o comando make(1) `all`, para criar as páginas web:

```
# cd en_US.ISO8859-1/htdocs
# make all
```

8.3. Instalando as web pages em seu Web Server

1. Se você tiver saído do diretório `en_US.ISO8859-1/htdocs`, volte para dele.

```
# cd /usr/build/en_US.ISO8859-1/htdocs
```

2. Execute o comando make(1) `install`, definindo a variável `DESTDIR` para o nome do diretório no qual deseja instalar os arquivos. Eles serão instalados no `$DESTDIR/data` o qual deve estar configurado para ser o diretório raiz do seu servidor web.

```
# env DESTDIR=/usr/local/www make install
```

3. Se você já instalou previamente as páginas web dentro deste mesmo diretório o processo de instalação não irá remover nenhuma página web antiga ou desatualizada. Por exemplo, se você compilar e instalar uma nova cópia do web site todos os dias, o comando abaixo irá procurar e remover todos os arquivos que não tenham sido alterados nos últimos três dias.

```
# find /usr/local/www -ctime 3 -print0 | xargs -0 rm
```

8.4. Variáveis de ambiente

ENGLISH_ONLY
: Se esta variável estiver definida e não for vazia, apenas a documentação em Inglês será compilada e instalada. Todas as traduções serão ignoradas. Por exemplo:

```
# make ENGLISH_ONLY=YES all install
```

Se você quiser desabilitar a variável ENGLISH_ONLY e compilar todas as páginas, incluindo traduções, basta definir a variável ENGLISH_ONLY para um valor vazio:

```
# make ENGLISH_ONLY="" all install clean
```

WEB_ONLY
: Se esta variável estiver definida e não for vazia, apenas as páginas HTML do diretório en_US.ISO8859-1/htdocs serão compiladas e instaladas. Todos os demais documentos do diretório en_US.ISO8859-1 (Handbook, FAQ, Tutorais, etc) serão ignorados. Por exemplo:

```
# make WEB_ONLY=YES all install
```

WEB_LANG
: Se esta variável estiver definida, apenas as páginas web no idioma especificado por ela serão compiladas e instaladas dentro do diretório /usr/build. Todos os demais idiomas serão ignorados. Por exemplo:

```
# make WEB_LANG="el_GR.ISO8859-7 es_ES.ISO8859-1 hu_HU.ISO8859-2 ↺
nl_NL.ISO8859-1" all install
```

NOPORTSCVS
: Se esta variável estiver definida, o processo de compilação não fará o check out dos arquivos do ports do repositório cvs. Ao invés disso, ele irá copiar os arquivos a partir do /usr/ports (ou do local definido na variável PORTSBASE).

WEB_ONLY, WEB_LANG, ENGLISH_ONLY e NOPORTSCVS são variáveis do make. Você pode definir estas variáveis no /etc/make.conf, no Makefile.inc, ou ainda como variáveis de ambiente na linha de comando ou nos arquivos de inicialização do seu *shell*.

Chapter 9. Traduções

Este é o FAQ para as pessoas que traduzem a documentação do FreeBSD (FAQ, Manual do FreeBSD, tutoriais, páginas de manual e outros) para diferentes línguas.

Ele é *fortemente* baseado na tradução do FAQ do Projeto Alemão de Documentação do FreeBSD, originalmente escrito por Frank Gründer <elwood@mc5sys.in-berlin.de > e traduzido novamente para o inglês por Bernd Warken <bwarken@mayn.de >.

Este FAQ é mantido pela Equipe de Engenharia de Documentação <doceng@FreeBSD.org >.

Q: Porque um FAQ?

A: Mais e mais pessoas estão se juntando à lista de discussão FreeBSD-doc e estão se oferecendo para traduzir a documentação do FreeBSD para outras línguas. Este FAQ visa responder as suas perguntas, de forma que eles possam iniciar a tradução da documentação o quanto antes.

Q: O que significa i18n e l10n

A: i18n significa internacionalização e l10n significa locallização. São apenas abreviações.

i18n pode ser lido como “i” seguido por 18 letras, seguidas por “n”. Similarmente, l10n é “l” seguido por 10 letras, seguidas por “n”.

Q: Existe uma lista de discussão para tradutores?

A: Sim. Os diferentes grupos de tradução possuem as suas próprias listas de discussão. A lista dos projetos de tradução possui maiores informações sobre as listas de discussão e sobre os web sites mantidos por cada um dos projetos de tradução.

Q: São necessários mais tradutores?

A: Sim. Quanto maior o número de pessoas trabalhando na tradução, mais rapidamente 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.

Q: Quais idiomas eu preciso conhecer?

A: Idealmente, você deverá possuir bons conhecimentos de Inglês escrito, e obviamente necessitará ser fluente na língua para a qual estiver traduzindo.

O conhecimento do idioma Inglês não é estritamente necessário. Por exemplo, você poderia fazer uma tradução do FAQ para o idioma Húngaro a partir da versão em Espanhol do documento.

Q: Quais softwares eu preciso conhecer?

A: É fortemente recomendado que você mantenha uma cópia local do repositório Subversion do FreeBSD (ao menos da parte referente a documentação). Isto pode ser feito executando o comando:

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

Note

Você irá precisar ter o devel/subversion 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 -r33733:33734 en_US.ISO8859-1/books/fdp-primer/book.ↄ
xml
```

Q: Como eu faço para descobrir se já existem outras pessoas traduzindo documentos para o meu idioma?

A: A página do Projeto de Tradução da Documentação lista os trabalhos de tradução que são conhecidos atualmente. Se outros já estão trabalhando na tradução da documentação para o seu idioma, por favor, não duplique os esforços. Ao invés disso, faça contato com o grupo para ver como pode ajudá-los.

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.

Q: Ninguém mais está traduzindo para o meu idioma. O que eu faço?

A: Parabés, você acabou de começar o “FreeBSD *`sua-língua-aqui`* Documentation Translation Project”. 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 `FAQ` ou um dos tutoriais.

Q: Eu já tenho alguns documentos traduzidos, para onde eu devo enviá-los?

A: 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).

Q: Eu sou a única pessoa trabalhando na tradução para este idioma, como faço para enviar meus documentos?

ou

Nós somos uma equipe de tradução, e queremos submeter os documentos que nossos membros traduziram para nós.

A: Primeiramente certifique-se que sua tradução está organizada corretamente. Isto significa que ela deve se encaixar na árvore de diretórios corrente e ser processada de maneira correta.

Atualmente a documentação do FreeBSD é armazenada em um diretório de nível superior chamado `head/`. Os diretórios abaixo deste são nomeados de acordo com o código do idioma em que eles estão escritos, definidos na ISO639 (`/usr/share/misc/iso639` em uma versão do FreeBSD mais nova que 20 de janeiro de 1999).

Se seu idioma puder ser codificado de maneiras diferentes (por exemplo, Chinês) então deverão existir outros diretórios abaixo deste, um para cada formato que você tenha forncecido.

Finalmente você deve ter diretórios para cada original.

Por exemplo, em uma hipotética tradução para o Sueco ficaria assim:

```
head/
    sv_SE.ISO8859-1/
                    Makefile
       htdocs/
```

```
docproj
                books/
                      faq/
                          Makefile
                          book.xml
```

`sv_SE.ISO8859-1` é o nome da tradução, na forma `lang.encoding` . Repare nos dois Makefiles que serão usados para construir a documentação.

Use tar(1) e gzip(1) para compactar sua documentação, e envie para o projeto.

```
% cd doc
% tar cf swedish-docs.tar sv_SE.ISO8859-1
% gzip -9 swedish-docs.tar
```

Coloque o arquivo `swedish-docs.tar.gz` em algum lugar. Se você não tiver acesso ao seu próprio espaço web (talvez seu ISP não disponibilize um), então você pode enviar um email para Equipe de Engenharia de Documentação <doceng@FreeBSD.org >, e combinar de enviar os arquivos por email quando for conveniente.

De qualquer forma, você deve usar o send-pr(1) para enviar um relatório indicando que você submeteu a documentação. Seria muito útil se você conseguisse outras pessoas para revisar a sua tradução antes de submetê-la, uma vez que é improvável que a pessoa que irá disponibilizá-la no repositório seja fluente no seu idoma.

Alguém (provavelmente o gerente do projeto de documentação, atualmente Equipe de Engenharia de Documentação <doceng@FreeBSD.org >) irá examinar os seus arquivos e irá confirmar se eles compilam sem erros. Em especial, os seguintes items serão verificados:

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, a pessoa que estiver examinando a sua submissão irá entrar em contato para que você faça as correções.

Se não exitir nenhum problema, os seus documentos traduzidos serão disponibilizados no repositório o quanto antes.

Q: Posso incluir uma língua ou um texto específico do país em minha tradução?

A: Nós preferimos que você não faça isso.

Por exemplo, suponha que você esteja traduzindo o Manual do FreeBSD para o Coreano e queira incluir uma seção sobre varejistas na Coréia em seu Manual do FreeBSD.

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 tem uma informação específica do seu país, por favor, submeta ela para alteração no Manual do FreeBSD em Inglês (usando send-pr(1)) e depois traduza novamente para sua língua no Manual do FreeBSD traduzido.

Obrigado.

Q: Como os caracteres específicos do idioma podem ser incluídos?

A: Caracteres não-ASCII devem ser incluídos na documentação usando entidades SGML.

Resumidamente, eles são um ``e'' comercial (&), o nome da entidade e um ponto-e-vírgula (;).

Os nomes de entidades estão definidos no ISO8879, que está na árvore do ports como 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 que você instalar o pacote iso8879, os arquivos em `/usr/local/share/xml/iso8879` conterão a lista completa.

Q: Dirigindo-se ao leitor

A: 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 polida.

Q: Eu preciso colocar alguma informação adicional nas minhas traduções?

A: Sim.

O cabeçalho da versão em Inglês de cada documento será parecido com o texto exibido abaixo:

```
<!--
     The FreeBSD Documentation Project

     $FreeBSD: head/en_US.ISO8859-1/books/faq/book.xml 38674 ↺
2012-04-14 13:52:52Z $
-->
```

A forma exata pode mudar, mas ela sempre incluirá a linha $FreeBSD$ e a frase `The FreeBSD Documentation Project`. Note que o $FreeBSD é inserido pelo svn, portanto ele deve estar vazio (apenas `$FreeBSD$`) para novos documentos.

O seu documento traduzido deve incluir sua própria linha $FreeBSD$, e você deve 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 deste arquivo pode começar com:

```
<!--
     The FreeBSD Spanish Documentation Project
     $FreeBSD: head/es_ES.ISO8859-1/books/faq/book.xml 38826 ↺
2012-05-17 19:12:14Z hrs $
     Original revision: r38674
     -->
```

Chapter 10. Estilo de Escrita

A fim de promover a consistência entre os inúmeros autores da documentação do FreeBSD, foram criadas algumas diretrizes para que os autores sigam.

Use a Grafia Inglesa Americana
: Existem diversas variantes do inglês, com grafias diferentes para a mesma palavra. Onde as grafias diferem, use a variante inglesa americana. Por exemplo: “color”, não “colour”, “rationalize”, não “rationalise”, e assim por diante.

Note

O uso do Inglês Britânico pode ser aceito para a contribuição de artigos, contudo a ortografia deverá ser consistente ao longo de todo o documento. Os outros documentos, tais como livros, web sites, páginas de manual, etc. deverão utilizar o Inglês Americano.

Não use contrações
: Não use contrações. Soletre sempre a frase por completo. A frase “ *Don't use contractions* ” seria errada.

  Evite fazer uso das contrações para obter um tom mais formal, será mais preciso, e ligeiramente mais fácil para os tradutores.

Use a vírgula de série
: Em uma lista de artigos dentro de um parágrafo, separe cada artigo do outro com uma vírgula. Separe o último artigo do outro com uma vírgula e a palavra “e”.

  Por o exemplo, observe o seguinte:

  > Esta é uma lista de um, dois e três artigos.

  Isto é uma lista de três artigos, “um”, “dois”, e “três”, ou de uma lista de dois artigos, “um” e “dois e três”?

  É melhor ser explícito e incluir uma vírgula de série:

  > Esta é uma lista de um, dois, e três artigos.

Evite frases redundantes
: Tente não usar frases redundantes. No detalhe, “o comando”, “o arquivo”, e “o comando man” são provavelmente redundantes.

Estes dois exemplos mostram isto para comandos. O segundo exemplo é preferido.

Use o comando `cvsup` para atualizar seus fontes.

Use o `cvsup` para atualizar seus fontes.

Estes dois exemplos mostram isto para nomes de arquivo. O segundo exemplo é preferido.

... no arquivo `/etc/rc.local` ...

... no `/etc/rc.local` ...

Estes dois exemplos mostram isto para referências aos manuais. O segundo exemplo é preferido (o segundo exemplo usa `citerefentry`).

Veja o `man csh` para maiores informações

veja o csh(1)

Dois espaços no final das sentenças.
Use sempre dois espaços no fim das sentenças, isto melhora a legibilidade, e facilita o uso de ferramentas tais como o Emacs.

Lembre-se que uma letra em caixa alta depois de um período, nem sempre denota uma sentença nova, especialmente na grafia de nomes. “Jordan K. Hubbard” é um bom exemplo; tem um `H` em caixa alta depois de um período e um espaço, e não há certamente uma sentença nova lá.

Para maiores informações sobre estilo de escrita, consulte Elementos de Estilo, por William Strunk.

10.1. Guia de Estilo

Para manter o código fonte da documentação consistente quando muitas pessoas diferentes o estão editando, por favor siga estas convenções de estilo.

10.1.1. Caixa de Letra (Maiúscula/Minúscula)

As `tags` devem ser utilizadas em caixa baixa, `<para>`, *não* `<PARA>`.

O texto que aparece em contextos do SGML é escrito geralmente em caixa alta, `<!ENTITY...>`, e `<!DOCTYPE...>`, *não* `<!entity...>` e `<!doctype...>`.

10.1.2. Acrônimos

Um acrônimo normalmente deve ser soletrado na primeira vez que ele aparecer em um livro, como por exemplo: "Network Time Protocol (NTP)". Depois que um acrônimo for

definido, você deveria passar a utilizar apenas ele (e não o termo completo, a não ser que isso faça mais sentido contextualmente). Normalmente um acrônimo é definido uma única vez por livro. Mas se você preferir, você também pode definí-lo quando ele aparecer pela primeira vez em cada capítulo.

Nas três primeiras vezes que um acrônimo for utilizado ele deve ser destacado com a tag <acronym> utilizando o atributo `role` setado com o termo completo como seu valor. Isto irá permitir que o link para o glossário seja criado, e habilitará um *mouseover* que quando renderizado irá exibir o termo completo.

10.1.3. Identação

Cada arquivo começa com a identação ajustada na coluna 0, *independentemente* do nível de identação do arquivo que pode vir a incluí-lo.

As tags de abertura aumentam o nível de identação em 2 espaços, as tags de fechamento diminuem o nível de identação em 2 espaços. Blocos de 8 espaços no começo de uma linha devem ser subistituidos por um Tab. Não use espaços na frente dos Tabs, e não adicione espaços em branco no final de uma linha. O conteúdo dentro de um elemento deve ser identado por dois espaços caso ele ocupe mais de uma linha.

Por exemplo, o código para esta seção seria algo como:

```
+--- This is column 0
V
<chapter>
  <title>...</title>

  <sect1>
    <title>...</title>

    <sect2>
      <title>Indentation</title>

      <para>Each file starts with indentation set at column 0,
 <emphasis>regardless</emphasis> of the indentation level of the file
 which might contain this one.</para>
      ...
    </sect2>
  </sect1>
</chapter>
```

Se você usar o Emacs ou XEmacs para editar os arquivos o `sgml-mode` será carregado automaticamente, e as variáveis locais do Emacs ao final de cada arquivo devem reforçar estes estilos.

Os usuários do Vim podem querer configurar o seu editor da seguinte forma:

```
augroup sgmledit
```

```
  autocmd FileType sgml set formatoptions=cq2l " Special formatting ↺
options
  autocmd FileType sgml set textwidth=70       " Wrap lines at 70 ↺
spaces
  autocmd FileType sgml set shiftwidth=2       " Automatically ↺
indent
  autocmd FileType sgml set softtabstop=2      " Tab key indents 2 ↺
spaces
  autocmd FileType sgml set tabstop=8          " Replace 8 spaces ↺
with a tab
  autocmd FileType sgml set autoindent         " Automatic ↺
indentation
augroup END
```

10.1.4. Estilo de Tags

10.1.4.1. Espaçamento de Tags

Tags que começam na mesma identação que um Tag precedente devem ser separados por uma linha em branco, e aqueles que não estão na mesma identação que um Tag precedente não, observe:

```
<article>
  <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>
```

10.1.4.2. Separando Tags

Tags tais como `itemizedlist` que terão sempre algum Tag adicional dentro dele, e que não fazem exame dos dados eles mesmos, estarão sempre sozinhos em uma linha.

Tags tais como `para` e `term` não necessitam outros Tags para conter caracteres normais, e o seu conteúdo começa imediatamente depois do Tag, *na mesma linha*.

O mesmo se aplica quando estes dois tipos de Tag se fecham.

Isto conduz a um problema óbvio ao misturar estes Tags.

Quando um Tag de abertura que não pode conter caracteres normais é utilizado logo após um Tag do tipo que requer outros Tags dentro dele para conter caracteres normais, eles devem estar em linhas separadas e o segundo Tag deve ser corretamente identado.

Quando um Tag que possa conter caracteres normais se fecha imediatamente depois que um Tag que não pode conter caracteres normais se fecha, eles podem coexistir na mesma linha.

10.1.5. Mudanças nos espaços em branco.

Ao enviar mudanças, não envie mudanças de conteúdo ao mesmo tempo que mudanças no formato.

Desta forma as equipes que convertem o manual para outras línguas podem ver rapidamente o que de fato mudou no conteúdo com o seu envio, sem ter que se decidir se uma linha mudou por causa do conteúdo, ou apenas porque foi reformatada.

Por exemplo, se você adicionar duas sentenças a um parágrafo, de forma que o comprimento das linhas no mesmo agora excedem 80 colunas, envie sua mudança sem corrigir o comprimento da linha. Ajuste então a quebra de linha e envie uma segunda mudança. Na mensagem de `commit` da segunda mudança, deixe claro que se trata apenas de um ajuste de formatação, e que a equipe da tradução pode ignorá-la.

10.1.6. Nonbreaking space

Evite as quebras de linha nos locais onde elas ficarem feias ou onde dificultarem a compreensão de uma sentença. As quebras de linha dependem da largura do meio de saída escolhido. Em particular, visualizar um documento em HTML com um navegador em modo texto pode conduzir a parágrafos mal formatados como o exemplo a seguir:

```
Data capacity ranges from 40 MB to 15
GB.  Hardware compression …
```

A entidade geral ` ` proibe a quebra de linha entre partes que devem permanecer juntas. Utilize *nonbreaking spaces* nos seguintes lugares:

- Entre números e suas unidades:

```
57600 bps
```

- Entre os nomes dos programas e os seus números de versão:

```
FreeBSD&amp;nbsp;4.7
```

- Entre nomes compostos (utilize com cuidado quando estiver lidando com nomes com mais de 3 ou 4 palavras, como por exemplo, “The FreeBSD Brazilian Portuguese Documentation Project”):

```
Sun&amp;nbsp;Microsystems
```

10.2. Lista de Palavras

O que se segue é uma pequena lista das palavras com a grafia da maneira que deve ser usada no projeto da documentação do FreeBSD. Se a palavra que você está procurando não estiver nesta lista, por favor consulte a lista de palavras da O'Reilly.

- 2.2.X
- 4.X-STABLE
- CD-ROM
- DoS *(Denial of Service)*
- Ports Collection
- IPsec
- Internet
- MHz
- Soft Updates
- Unix
- disk label
- email
- file system
- manual page
- mail server
- name server

- null-modem
- web server

Chapter 11. Usando `sgml-mode` com o Emacs

As versões recentes do Emacs ou XEmacs (disponível na coleção dos `ports`) contêm um pacote muito útil chamado PSGML (ele pode ser instalado pelo editors/psgml). Ele é automaticamente invocado quando um arquivo com a extensão `.xml` é carregado, ou executando `M-x sgml-mode` , ele é a modalidade principal para tratar dos elementos e dos atributos de um arquivo SGML.

Compreender alguns dos comandos fornecidos por esta modalidade pode tornar o trabalho com os documentos em SGML, tais como o Handbook, muito mais fácil.

`C-c C-e`
: Executa o `sgml-insert-element` . Você será questionado sobre o nome do elemento que deseja inserir no ponto atual. Você pode usar a tecla TAB para completar o elemento. Os elementos que são inválidos no ponto atual não serão permitidos.

  As tags de abertura e de fechamento para o elemento serão inseridas. Se o elemento contiver outro, obrigatórios, os elementos destes também serão inseridos.

`C-c =`
: Executa o `sgml-change-element-name` . Coloque o cursor dentro de um elemento e execute este comando. Você será questionado sobre o nome do elemento para o qual você deseja mudar. As tags de abertura e de fechamento do elemento atual serão alterados para o novo elemento.

`C-c C-r`
: Executa o `sgml-tag-region` . Selecione algum texto (posicione o cursor no começo do texto, de `C-espaço`, agora posicione o cursor no final do texto, de `C-espaço`) e execute então este comando. Você será questionado sobre o nome do elemento que deseja inserir. Este elemento será inserido então imediatamente antes e depois da região marcada.

`C-c -`
: Executa o `sgml-untag-element` . Coloque o cursor dentro da tag de abertura ou de fechamento do elemento que você quer remover, e execute este comando. As tags de abertura ou de fechamento do elemento serão removidas.

`C-c C-q`
: Executa o `sgml-fill-element` . Irá reformatar recursivamente o elemento atual. A reformatação *afetará* o conteúdo em que os espaços em branco são significativos, como dentro de elementos `programlisting`, assim utilize este comando com cuidado.

`C-c C-a`
: Executa o `sgml-edit-attributes` . Abre um segundo `buffer` que contém uma lista de todos os atributos e valores atuais para o elemento mais próximo. Use a tecla TAB

para navegar entre os atributos, utilize `C-k` para remover um valor existente e para substituí-lo com um novo, utilize `C-c C-c` para fechar este `buffer` e para retornar ao documento principal.

`C-c C-v`
: Executa o `sgml-validate`. Você será questionado se deseja salvar o documento atual (se necessário) e então executa uma validação do SGML. A saída da validação é capturada em um novo `buffer`, e você poderá então navegar de um foco de problema para outro, corrigindo os erros de marcação durante este processo.

`C-c /`
: Executa `sgml-insert-end-tag` . Insere a tag de fechamento para o elemento atual que está aberto.

Sem dúvida há outras funções úteis desta modalidade, mas estas são as que você irá utilizar com mais frequência.

Você também pode usar as seguintes entradas no `.emacs` para ajustar o espaçamento apropriado, a identação, e a largura de coluna para trabalhar com o projeto de documentação.

```
    (defun local-sgml-mode-hook
      (setq fill-column 70
            indent-tabs-mode nil
            next-line-add-newlines nil
            standard-indent 4
            sgml-indent-data t)
      (auto-fill-mode t)
      (setq sgml-catalog-files '("/usr/local/share/xml/catalog")))
    (add-hook 'psgml-mode-hook
      '(lambda () (local-psgml-mode-hook)))
```

Chapter 12. Veja também

Este documento não é deliberadamente uma discussão exaustiva sobre SGML, DTDs, ou do projeto de documentação do FreeBSD. Para obter maiores informações, sugerimos que você visite os seguintes sítios web.

12.1. Projeto de documentação do FreeBSD

- Páginas web do Projeto de documentação do FreeBSD
- Manual do FreeBSD

12.2. SGML

- Página web sobre SGML/XML, referências detalhadas sobre SGML
- Uma breve introdução ao SGML

12.3. HTML

- Consórcio `World Wide Web`
- As especificações do HTML 4.0

12.4. DocBook

- O comitê técnico do DocBook, responsáveis pelo DTD DocBook.
- DocBook: O guia definitivo, documentação online para o DTD DocBook.
- Repositório aberto de DocBook contém folhas de estilo DSSSL e outros recursos para pessoas que utilizam DocBook.

12.5. Projeto de documentação do Linux

- Pagínas web do projeto de documentação do Linux

Appendix A. Exemplos

Este apêndice contém arquivos SGML de exemplo e linhas de comando que você pode utilizar para convertê-los de um formato para outro. Se você instalou com sucesso as ferramentas do Projeto de Documentação, então você será capaz de utilizar estes exemplos imediatamente.

Estes exemplos não são exaustivos — eles não contêm todos os elementos que você pode utilizar, particularmente para a capa do seu documento. Para maiores exemplos de marcação DocBook você deve examinar o código SGML deste e de outros documentos, disponíveis no repositório `doc` do svn, ou disponíveis online no endereço `http://svnweb.FreeBSD.org/doc/` .

Para evitar confusão, estes exemplos utilizam a especificação DocBook 4.1 DTD sem nenhuma extensão particular adicionada pelo FreeBSD. Eles também utilizam as folhas de estilo padrões distribuídas pelo Norm Walsh, sem nenhuma customização feita nas mesmas pelo Projeto de Documentação do FreeBSD. Isto os torna mais úteis como exemplos genéricos de marcação DocBook.

A.1. DockBook **book**

Example A.1. DocBook **book**

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<book>
  <bookinfo>
    <title>Um exemplo de Livro</title>

    <author>
      <firstname>Seu nome</firstname>
      <surname>Seu sobrenome</surname>
      <affiliation>
        <address><email>seuemail@example.com</email></address>
      </affiliation>
    </author>

    <copyright>
      <year>2000</year>
      <holder>Texto de Copyright</holder>
    </copyright>

    <abstract>
      <para>Se seu livro possui um sumário ele deve ser ↺
colocado aqui.</para>
    </abstract>
```

```
  </bookinfo>

  <preface>
    <title>Prefácio</title>

    <para>Seu livro pode ter um prefácio, se este for o caso, ↺
ele deve
      ser colocado aqui.</para>
  </preface>

  <chapter>
    <title>Meu primeiro capítulo</title>

    <para>Este é o primeiro capítulo do meu livro.</para>

    <sect1>
      <title>Minha primeira seção</title>

      <para>Esta é a primeira seção do meu livro.</para>
    </sect1>
  </chapter>
</book>
```

A.2. DocBook `article`

Example A.2. DocBook `article`

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V4.1//EN">

<article>
  <articleinfo>
    <title>Um exemplo de Artigo</title>

    <author>
      <firstname>Seu nome</firstname>
      <surname>Seu sobrenome</surname>
      <affiliation>
        <address><email>seuemail@example.com</email></address>
      </affiliation>
    </author>

    <copyright>
      <year>2000</year>
      <holder>Texto de Copyright</holder>
    </copyright>

    <abstract>
```

```
      <para>Se o seu artigo possuir um sumário ele deve ser ↺
colocado aqui.</para>
    </abstract>
  </articleinfo>

  <sect1>
    <title>Minha primeira seção</title>

    <para>Esta é a primeira seção do meu artigo.</para>

    <sect2>
      <title>Minha primeira subseção</title>

      <para>Esta é a primeira subseção do meu artigo.</para>
    </sect2>
  </sect1>
</article>
```

A.3. Produzindo saídas formatadas

Esta seção assume que você já instalou os softwares listados no port textproc/docproj, seja via meta-port ou manualmente. Além disso, ela também assume que os seus softwares estão instalados em subdiretórios sob o `/usr/local/`, e que os diretórios nos quais os binários foram instalados, estão mapeados no seu `PATH`. Ajuste os paths conforme a necessidade do seu sistema.

A.3.1. Usando o Jade

Example A.3. Convertendo de DocBook para HTML (em um único grande arquivo)

```
% jade -V nochunks \   ❶
    -c /usr/local/share/xml/docbook/dsssl/modular/catalog \    ❷
    -c /usr/local/share/xml/docbook/catalog \
    -c /usr/local/share/xml/jade/catalog \
    -d /usr/local/share/xml/docbook/dsssl/modular/html/docbook.↺
dsl \❸
    -t sgml ❹ file.xml > file.html  ❺
```

❶ Especifique o parâmetro `nochunks` para as folhas de estilo, forçando que todos os outputs sejam escritos para a saída padrão (STDOUT) (utilizando as folhas de estilo do Norm Walsh).

❷ Especifique os catálogos que o Jade terá que processar. Três catálogos são requeridos. O primeiro é o catálogo que contém as informações sobre as fol-

has de estilo DSSSL. O segundo contém informações sobre o DTD DockBook. E o terceiro contém informações específicas para o Jade.

❸ Especifique o caminho completo das folhas de estilo DSSSL as quais o Jade irá utilizar quando estiver processando o documento.

❹ Instrua o Jade para realizar uma *transformação* de uma DTD para outra. Neste caso, a entrada será transformada de um DTD DocBook para um DTD HTML.

❺ Especifique o arquivo que o Jade deve processar, e redirecione a saída para o arquivo `.html` desejado.

Example A.4. Convertendo de DocBook para HTML (vários arquivos pequenos)

```
% jade \
    -c /usr/local/share/xml/docbook/dsssl/modular/catalog \    ❶
    -c /usr/local/share/xml/docbook/catalog \
    -c /usr/local/share/xml/jade/catalog \
    -d /usr/local/share/xml/docbook/dsssl/modular/html/docbook.↺
dsl \❷
    -t sgml ❸ file.xml ❹
```

❶ Especifique os catálogos os quais o Jade terá que processar. Três catálogos são requeridos. O primeiro é o catálogo o qual contém as informações sobre as folhas de estilo DSSSL. O segundo contém informações sobre o DTD DocBook. O terceiro contém informações específicas para o Jade.

❷ Especifique o caminho completo da folha de estilo DSSSL a qual o Jade irá utilizar quando estiver processando o documento.

❸ Instrua o Jade para realizar a *transformação* de uma DTD para outra. Neste caso, a entrada será transformada de um DTD DocBook para um DTD HTML.

❹ Especifique o arquivo que o Jade deve processar. A folha de estilo determina como os arquivos HTML individuais serão nomeados, inclusive o nome do arquivo "raiz" (é o arquivo que contém o inicio do documento).

Este exemplo pode continuar gerando apenas um único arquivo HTML, dependerá da estrutura do documento que você estiver processando e das regras da folha de estilo selecionada, para divisão do output.

Example A.5. Convertendo de DocBook para Postscript

O arquivo fonte SGML precisa ser convertido para um arquivo TeX.

```
% jade -V tex-backend \ ❶
    -c /usr/local/share/xml/docbook/dsssl/modular/catalog \    ❷
    -c /usr/local/share/xml/docbook/catalog \
    -c /usr/local/share/xml/jade/catalog \
    -d /usr/local/share/xml/docbook/dsssl/modular/print/
docbook.dsl \❸
    -t tex ❹ file.xml
```

❶ Customize as folhas de estilo para utilizar as várias opções existentes, específicas para a produção de saídas TeX.

❷ Especifique os catálogos os quais o Jade terá que processar. Três catálogos são requeridos. O primeiro é o catálogo o qual contém as informações sobre as folhas de estilo DSSSL. O segundo contém informações sobre o DTD DocBook. O terceiro contém informações específicas para o Jade.

❸ Especifique o caminho completo da folha de estilo DSSSL a qual o Jade irá utilizar quando estiver processando o documento.

❹ Instrua o Jade para converter o output para TeX.

O arquivo `.tex` gerado, deve ser agora processado pelo `tex`, especificando o pacote de macros `&jadetex`.

```
% tex "&jadetex" file.tex
```

Você tem que executar o `tex` *pelo menos* três vezes. A primeira execução irá processar o documento, e determinar as áreas do documento que são referenciadas a partir de outras partes do documento, para uso na indexação, etc.

Não fique alarmado se você visualizar mensagens de alertas tais como LaTeX Warning: Reference `136' on page 5 undefined on input line 728. neste momento.

A segunda execução reprocessa o documento agora que certas peças de informação são conhecidas (tais como o número de páginas do documento). Isto permite indexar as entradas e estabelecer as outras referências cruzadas.

A terceira execução irá realizar a limpeza final necessária no arquivo

O output deste estágio será um `arquivo.dvi` .

Finalmente, execute o `dvips` para converter o arquivo `.dvi` para o formato Postscript.

```
% dvips -o file.ps file.dvi
```

Example A.6. Convertendo de DocBook para PDF

A primeira parte deste processo é idêntica ao realizado quando se converte de DocBook para Postscript, utilizando a mesma linha de comando para o `jade` (Example A.5, “Convertendo de DocBook para Postscript”).

Quando o arquivo `.tex` já tiver sido gerado, você deve executar o pdfTeX utilizando o pacote de macros `&pdfjadetex`.

```
% pdftex "&pdfjadetex" file.tex
```

De novo, execute este comando três vezes.

Ele irá gerar um `arquivo .pdf`, o qual não necessita de nenhum processamento adicional.

Index