Guia dos Committers

Resumo

Este documento fornece informações para a comunidade de committers do FreeBSD. Todos os novos committers devem ler este documento antes de começar, e os committers existentes são fortemente encorajados a revisá-lo de tempos em tempos.

Quase todos os desenvolvedores do FreeBSD possuem direitos de commit para um ou mais repositórios. No entanto, alguns desenvolvedores não tem, e algumas das informações aqui se aplicam a eles também. (Por exemplo, algumas pessoas só têm direitos para trabalhar com o banco de dados do Relatório de Problemas). Por favor, veja Problemas Específicos para Desenvolvedores Que Não São Committers para mais informações.

Este documento também pode ser de interesse para os membros da comunidade FreeBSD que querem aprender mais sobre como o projeto funciona.

Índice

1. Detalhes Administrativos . . . 2
2. Chaves OpenPGP para o FreeBSD . . . 3
3. Kerberos e LDAP Web Password para o cluster do FreeBSD . . . 4
4. Tipos de Commit Bits . . . 5
5. Subversion Primer . . . 6
6. Configuração, Convenções e Tradições . . . 27
7. Revisão pré-commit . . . 31
8. Mensagens de Log de Commit . . . 32
9. Licença preferida para novos arquivos . . . 36
10. Acompanhando as Licenças Concedidas ao Projeto FreeBSD . . . 37
11. Relações entre os Desenvolvedores . . . 38
12. Se em dúvida... . . . 38
13. Bugzilla . . . 39
14. Phabricator . . . 39
15. Quem é quem . . . 40
16. Guia de início rápido do SSH . . . 41
17. Disponibilidade do Coverity® para os Committers do FreeBSD . . . 42
18. A Grande Lista de Regras dos Committers do FreeBSD . . . 42
19. Suporte para Várias Arquiteturas . . . 50
20. FAQ específico dos Ports . . . 54
21. Problemas Específicos para Desenvolvedores Que Não São Committers . . . 55
22. Perguntas Diversas . . . 56

1. Detalhes Administrativos

Métodos de login	ssh(1),_apenas no protocolo 2
Servidor Principal de Shell	`freefall.FreeBSD.org`
Servidor SMTP	`smtp.FreeBSD.org:587` (veja também Configuração de acesso SMTP).
Raiz do Subversion para o `src/`	`svn+ssh://repo.FreeBSD.org/base` (veja também Branches `RELENG_*` e Layout Geral).
Raiz do Subversion para o `doc/`	`svn+ssh://repo.FreeBSD.org/doc` (veja também Branches e Layout do Projeto de Documentação do FreeBSD).
Raiz do Subversion para o `ports/`	`svn+ssh://repo.FreeBSD.org/ports` (veja também Branches e Layout da Árvore de Ports do FreeBSD).
Listas de Discussão Internas	developers (tecnicamente chamada de all-developers), doc-developers, doc-committers, ports-developers, ports-committers, src-developers, src-committers. (cada repositório do projeto tem as suas próprias listas de discussão -developers e -committers. Os arquivos destas listas podem ser encontrados nos arquivos /local/mail/repository-name-developers-archive e /local/mail/repository-name-committers-archive no cluster `FreeBSD.org`.)
Relatórios mensais do Core Team	Disponíveis no diretório /home/core/public/monthly-reports do cluster `FreeBSD.org`.
Relatórios mensais da Equipe de Gestão do Ports	Disponíveis no diretório /home/portmgr/public/monthly-reports do cluster `FreeBSD.org`.
Branchs SVN do `src/` dignas de atenção	`stable/n` (`n`-STABLE), `head` (-CURRENT)

O ssh(1) é necessário para se conectar aos servidores do projeto. Para mais informações, veja Guia de início rápido do SSH.

Links Úteis:

- Páginas Internas do Projeto FreeBSD
- Servidores do Projeto FreeBSD
- Grupos Administrativos do Projeto FreeBSD

2. Chaves OpenPGP para o FreeBSD

O projeto FreeBSD utiliza chaves criptográficas em conformidade com o padrão OpenPGP (*Pretty Good Privacy*) para autenticar os committers. Mensagens contendo informações importantes como chaves públicas SSH podem ser assinadas com uma chave OpenPGP para provar que elas vieram realmente do committer. Veja PGP & GPG: Email for the Practical Paranoid by Michael Lucas e http://en.wikipedia.org/wiki/Pretty_Good_Privacy para maiores informações.

2.1. Criando uma chave

As chaves existentes podem ser usadas, mas elas devem ser obtidas com o doc/head/shared/pgpkeys/checkkey.sh primeiro. Neste caso, certifique-se que a chave contenha um ID de usuário FreeBSD.

Para quem ainda não tem uma chave OpenPGP, ou precisa de uma nova chave para atender aos requisitos de segurança do FreeBSD, nesta seção iremos mostrar como gerar uma.

1. Instale o security/gnupg. Digite estas linhas no ~/.gnupg/gpg.conf para definir padrões mínimos aceitáveis:

```
fixed-list-mode
keyid-format 0xlong
personal-digest-preferences SHA512 SHA384 SHA256 SHA224
default-preference-list SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 BZIP2
ZLIB ZIP Uncompressed
use-agent
verify-options show-uid-validity
list-options show-uid-validity
sig-notation issuer-fpr@notations.openpgp.fifthhorseman.net=%g
cert-digest-algo SHA512
```

2. Gere uma chave:

```
% gpg --full-gen-key
gpg (GnuPG) 2.1.8; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Warning: using insecure memory!
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048 ①
Requested keysize is 2048 bits
```

```
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 3y ②
Key expires at Wed Nov  4 17:20:20 2015 MST
Is this correct? (y/N) y
GnuPG needs to construct a user ID to identify your key.

Real name: Chucky Daemon ③
Email address: notreal@example.com
Comment:
You selected this USER-ID:
"Chucky Daemon <notreal@example.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.
```

① Chaves de 2048-bit com uma expiração de 3 anos proveem uma proteção adequada nos dias de hoje (2013-12). O artigo http://danielpocock.com/rsa-key-sizes-2048-or-4096-bits descreve a situação em mais detalhes.

② Três anos é um tempo de vida para a chave que é curto o suficiente para tornar obsoletas as chaves enfraquecidas pelo avanço do poder computacional, mas é tempo suficiente para reduzir os principais problemas de gerenciamento.

③ Use o seu nome real, preferencialmente o mesmo que consta do seu documento de identificação (ID) emitido pelo seu governo para tornar mais fácil para as outras pessoas confirmarem sua identidade. Você pode inserir um texto adicional para ajudar outras pessoas a identificar você na seção `Comment`.
Depois que o endereço de e-mail é inserido, uma senha é solicitada. Os métodos de criação de uma frase secreta segura são controversos. Em vez de sugerir um único caminho, aqui estão alguns links para sites que descrevem vários métodos: http://world.std.com/~reinhold/diceware.html, http://www.iusmentis.com/security/passphrasefaq/, http://xkcd.com/936/ e http://en.wikipedia.org/wiki/Passphrase.

Proteja a chave privada e a frase secreta. Se a chave privada ou a frase secreta pode ter sido comprometida ou exposta, notifique imediatamente accounts@FreeBSD.org e revogue a chave.

O processo para commit da nova chave é mostrado em Procedure: Etapas para os novos committers.

3. Kerberos e LDAP Web Password para o cluster do FreeBSD

O cluster do FreeBSD requer uma senha do Kerberos para acessar certos serviços. A senha do Kerberos também serve como a senha LDAP para a web, já que o LDAP faz um proxy para o

Kerberos no cluster. Alguns dos serviços que exigem isso incluem:

- Bugzilla
- Jenkins

Para criar uma nova conta do Kerberos no cluster do FreeBSD, ou para redefinir uma senha do Kerberos para uma conta existente usando um gerador de senha aleatória:

```
% ssh kpasswd.freebsd.org
```



Isto deve ser feito a partir de uma máquina fora do cluster do FreeBSD.org.

Uma senha do Kerberos também pode ser definida manualmente, para isto logue no servidor `freefall.FreeBSD.org` e execute:

```
% kpasswd
```



A menos que os serviços autenticados por Kerberos do cluster do FreeBSD.org tenham sido usados anteriormente, o erro `Client unknown` será exibido. Este erro significa que o método `ssh kpasswd.freebsd.org` mostrado acima deve ser usado primeiro para inicializar a conta Kerberos.

4. Tipos de Commit Bits

O repositório do FreeBSD possui um número de componentes que, quando combinados, suportam os fontes básicos do sistema operacional, a documentação, a infraestrutura de ports de aplicativos de terceiros e vários utilitários mantidos. Quando os commit bits do FreeBSD são alocados, as áreas da árvore onde o bit pode ser usado são especificadas. Geralmente, as áreas associadas a um bit refletem quem autorizou a alocação do bit de commit. Áreas adicionais de autoridade podem ser adicionadas em uma data posterior: quando isso ocorre, o committer deve seguir procedimentos normais de atribuição de bits de confirmação para essa área da árvore, buscando a aprovação da entidade apropriada e possivelmente obtendo um mentor para essa área por algum período de tempo.

Tipo de Committer	*Responsável*	*Componentes da Árvore*
src	core@	src/, doc/ sujeito a revisão apropriada
doc	doceng@	doc/, ports/, src/ documentação
ports	portmgr@	ports/

Os commit bits alocados antes do desenvolvimento da noção de áreas de autoridade podem ser apropriados para uso em muitas partes da árvore. No entanto, o senso comum determina que um committer que não tenha trabalhado anteriormente em uma área da árvore busque alguém para realizar uma revisão antes de realizar o commit, busque a aprovação da parte responsável

apropriada e/ou trabalhe com um mentor. Uma vez que as regras relativas à manutenção de código diferem por área da árvore, isso é tanto para o benefício do committer trabalhando em uma área de menos familiar quanto para as outras pessoas trabalhando na árvore.

Committers são encorajados a buscar revisão do seu trabalho como parte do processo normal de desenvolvimento, independentemente da área da árvore onde o trabalho está ocorrendo.

4.1. Política para atividade de Committers em outras árvores

- Todos os committers podem modificar base/head/shared/misc/committers-*.dot, base/head/usr.bin/calendar/calendars/calendar.freebsd, e ports/head/astro/xearth/files.
- os doc committers podem efetuar commits de alterações na documentação sob src, tal como man pages, READMEs, banco de dados do fortune, arquivos de calendários, e comentar correções sem aprovação de um src committer, sujeito aos cuidados normais necessários para um commit seguro.
- Qualquer committer pode fazer alterações em qualquer outra árvore com um "Approved by" de um committer que não esteja sob mentoria e que tenha o bit apropriado.
- Os committers podem adquirir um bit adicional pelo processo usual de encontrar um mentor que os proporá ao core, doceng ou portmgr, conforme apropriado. Quando aprovados, eles serão adicionados ao 'access' e o período normal de mentoria irá ocorrer, o que envolverá a continuidade do uso do "Approved by" por um período.
- O "Approved by" só é aceitável vindo de src committers que não estejam sob mentoria - os committers mentorados podem fornecer um "Reviewed by" mas não um "Approved by".

5. Subversion Primer

Supõe-se que novos committers já estejam familiarizados com a operação básica do Subversion. Se não, comece lendo o Subversion Book.

5.1. Introdução

O repositório de código fonte do FreeBSD mudou do `CVS` para o Subversion em 31 de maio de 2008. O primeiro commit real no `SVN` é o *r179447*.

O repositório do FreeBSD `doc/www` foi migrado do `CVS` para o Subversion em 19 de Maio de 2012. O primeiro commit real no `SVN` foi o *r38821*.

O repositório da coleção de `ports` do FreeBSD foi migrado do `CVS` para o Subversion em 14 de Julho 2012. O primeiro commit real no `SVN` foi o *r300894*.

O Subversion pode ser instalado a partir da Coleção de Ports do FreeBSD, executando o comando:

```
# pkg install subversion
```

5.2. Primeiros Passos

Existem algumas maneiras de obter uma cópia de trabalho da árvore do Subversion. Esta seção irá explicá-las.

5.2.1. Checkout direto

O primeiro é fazer check-out diretamente do repositório principal. Para a árvore `src`, use:

```
% svn checkout svn+ssh://repo.freebsd.org/base/head /usr/src
```

Para a árvore `doc`, use:

```
% svn checkout svn+ssh://repo.freebsd.org/doc/head /usr/doc
```

Para a árvore da coleção de `ports`, use:

```
% svn checkout svn+ssh://repo.freebsd.org/ports/head /usr/ports
```



Embora os exemplos restantes neste documento tenham sido escritos tendo o workflow de trabalho com a árvore `src` em mente, os conceitos básicos são os mesmos para se trabalhar com o `doc` e com a árvore dos `ports`. Os ports relacionados às operações do Subversion estão listados em FAQ específico dos Ports.

O comando acima irá fazer o checkout da árvore de código-fonte `CURRENT` como /usr/src/, o qual pode ser qualquer diretório de destino no sistema de arquivos local. Omitir o argumento final desse comando faz com que a cópia de trabalho, neste caso, seja nominada como "head", mas ela poderá ser renomeada com segurança.

O `svn+ssh` significa que o protocolo `SVN` será tunelado sobre o SSH. O nome do servidor é `repo.freebsd.org`, e `base` é o caminho para o repositório, e `head` é o subdiretório dentro do repositório.

Se seu nome de login no projeto FreeBSD for diferente do nome de login usado na sua máquina local, inclua-o na URL (por exemplo `svn+ssh://jarjar@repo.freebsd.org/base/head`) ou adicione uma entrada no arquivo ~/.ssh/config no formato:

```
Host repo.freebsd.org
    User jarjar
```

Este é o método mais simples, mas é difícil dizer quanta carga ele irá colocar no repositório.



O `svn diff` não requer acesso ao servidor pois o `SVN` armazena uma cópia de

> referência de todos os arquivos na cópia de trabalho. Isto, no entanto, significa que as cópias de trabalho do Subversion são muito grandes em tamanho.

5.2.2. Branches `RELENG_*` e Layout Geral

No `svn+ssh://repo.freebsd.org/base`, *base* refere-se à árvore de código fonte. Similarmente, *ports* refere-se à árvore de ports e assim por diante. Estes são repositórios separados com suas próprias seqüências de números de mudança, controles de acesso e email de commit.

Para o repositório base, HEAD refere-se a árvore -CURRENT. Por exemplo, head/bin/ls é o que entraria como /usr/src/bin/ls em uma release. Alguns locais importantes são:

- */head/* que corresponde a `HEAD`, também conhecido como `-CURRENT`.
- */stable/n* que corresponde a `RELENG_n_`.
- */releng/n.n* que corresponde a `RELENG_n_n`.
- */release/n.n.n* que corresponde a `RELENG_n_n_n_RELEASE`.
- */vendor** é uma área de trabalho para importar branches de vendors. Este diretório em si não contém branches, no entanto, os seus subdiretórios contém. Isso contrasta com os diretórios *stable*, *releng* e *release*.
- */projects* e */user* são áreas de trabalho de branch. Como acima, o diretório */user* não contém branches em si.

5.2.3. Branches e Layout do Projeto de Documentação do FreeBSD

no `svn+ssh://repo.freebsd.org/doc`, *doc* refere-se à raiz do repositório da árvore de código fonte.

Em geral, a maior parte do trabalho do Projeto de Documentação do FreeBSD será feito dentro do branch head/ da árvore com os arquivos fontes da documentação.

A documentação do FreeBSD é escrita e/ou traduzida para vários idiomas, cada um em um diretório separado no branch head/.

Cada conjunto de tradução contém vários subdiretórios para as várias partes do Projeto de Documentação do FreeBSD. Alguns diretórios notáveis são:

- */articles/* contém o código fonte para artigos escritos por vários colaboradores do FreeBSD.
- */books/* contém o código fonte para os diferentes livros, como o Handbook do FreeBSD.
- */htdocs/* contém o código-fonte do website do FreeBSD.

5.2.4. Branches e Layout da Árvore de Ports do FreeBSD

No `svn+ssh: //repo.freebsd.org/ports`, *ports* refere-se à raiz do repositório da árvore de ports.

Em geral, a maior parte do trabalho na coleção de ports do FreeBSD será feito dentro da branch head/ da árvore de ports, que é a árvore de ports real usada para instalar software. Alguns outros locais importantes são:

- /branches/RELENG_n_n_n que corresponde a `RELENG_n_n_n` e é usado para mesclar atualizações de segurança em preparação para uma release.
- /tags/RELEASE_n_n_n que corresponde a `RELEASE_n_n_n` e representa uma tag de release da árvore de ports.
- /tags/RELEASE_n_EOL representa o tag de end of life de um branch específico do FreeBSD.

5.3. Uso diário

Esta seção explicará como realizar operações comuns do dia-a-dia com o Subversion.

5.3.1. Ajuda

O `SVN` traz em si uma documentação interna de ajuda. Ela pode ser acessada digitando:

```
% svn help
```

Informações adicionais podem ser encontradas no Subversion Book.

5.3.2. Checkout

Como visto anteriormente, para fazer o checkout da branch principal (head) do FreeBSD:

```
% svn checkout svn+ssh://repo.freebsd.org/base/head /usr/src
```

Em algum momento, provavelmente será útil ter uma copia de trabalho mais do que apenas do `HEAD`, por exemplo, ao mesclar as alterações para stable/7. Portanto, pode ser útil ter uma verificação parcial da árvore completa (um check-out completo seria muito doloroso).

Para fazer isso, primeiro confira a raiz do repositório:

```
% svn checkout --depth=immediates svn+ssh://repo.freebsd.org/base
```

Isso vai obter o `base` com todos os arquivos que ele contém (no momento da escrita, apenas o ROADMAP.txt) e subdiretórios vazios para `head`, `stable`, `vendor` e assim por diante.

É possível expandir a cópia de trabalho. Basta alterar a profundidade dos vários subdiretórios:

```
% svn up --set-depth=infinity base/head
% svn up --set-depth=immediates base/release base/releng base/stable
```

O comando acima irá baixar uma cópia completa do `head`, além de cópias vazias de cada tag de `release`, de cada branch de `releng`, e de cada branch `stable`.

Se numa data posterior você tiver necessidade de mesclar algo, por exemplo, com a `7-STABLE`,

expanda a cópia de trabalho:

```
% svn up --set-depth=infinity base/stable/7
```

As subárvores não precisam ser expandidas completamente. Por exemplo, para expandir apenas o `stable/7/sys` e depois expandir o resto do `stable/7`:

```
% svn up --set-depth=infinity base/stable/7/sys
% svn up --set-depth=infinity base/stable/7
```

Atualizar a árvore com `svn update` irá apenas atualizar o que foi pedido anteriormente (neste caso, `head` e `stable/7`), ele não irá puxar a árvore inteira.

5.3.3. Checkout Anonimo

É possível efetuar o checkout anonimo o repositório Subversion do FreeBSD. Isso dará acesso a uma árvore somente leitura que pode ser atualizada, mas que não poderá ser enviada de volta para o repositório principal por meio de um commit. Para fazer isso, use:

```
% svn co https://svn.FreeBSD.org/base/head /usr/src
```

Mais detalhes sobre o uso do Subversion podem ser encontrados em Usando o Subversion.

5.3.4. Atualizando a Árvore

Para atualizar uma cópia de trabalho para a revisão mais recente ou uma revisão específica:

```
% svn update
% svn update -r12345
```

5.3.5. Status

Para ver as alterações locais que foram feitas na cópia de trabalho:

```
% svn status
```

Para mostrar alterações locais e arquivos que estão desatualizados, execute:

```
% svn status --show-updates
```

5.3.6. Editando e efetuando o commit

O SVN não precisa ser avisado com antecedência sobre a edição de arquivos.

Para efetuar o commit de todas as alterações no diretório atual e em todos os seus subdiretórios:

```
% svn commit
```

Para efetuar o commit de todas as alterações, por exemplo, nos arquivos lib/libfetch/ e usr/bin/fetch/ em uma única operação, execute:

```
% svn commit lib/libfetch usr/bin/fetch
```

Também existe um wrapper de commit para a árvore de ports para manipular as propriedades e para verificar a sanidade das mudanças:

```
% /usr/ports/Tools/scripts/psvn commit
```

5.3.7. Adicionando e Removendo Arquivos



Antes de adicionar arquivos, obtenha uma cópia do auto-props.txt (há também um versão específica da árvore de ports) e adicione-o ao ~/.subversion/config seguindo as instruções contidas no arquivo. Se você adicionou algo antes de ler isto, use o comando `svn rm --keep-local` para apenas os arquivos adicionados, corrija seu arquivo de configuração e adicione-os novamente. O arquivo de configuração inicial é criado quando você executa um comando svn pela primeira vez, até mesmo algo tão simples quanto `svn help`.

Os arquivos são adicionados a um SVN repositório com `svn add`. Para adicionar um arquivo chamado *foo*, edite-o e depois execute:

```
% svn add foo
```



A maioria dos novos arquivos fonte deve incluir uma string `$FreeBSD: head/pt_BR.ISO8859-1/articles/committers-guide/article.xml 53731 2020-01-01 14:37:53Z ebrandi $` próxima ao início do arquivo. Ao efetuar o commit, o svn expandirá a string `$FreeBSD: head/pt_BR.ISO8859-1/articles/committers-guide/article.xml 53731 2020-01-01 14:37:53Z ebrandi $`, adicionando o caminho do arquivo, o número da revisão, a data e a hora da confirmação e o nome de usuário do committer. Arquivos que não podem ser modificados podem ser enviados sem a string `$FreeBSD: head/pt_BR.ISO8859-1/articles/committers-guide/article.xml 53731 2020-01-01 14:37:53Z ebrandi $`.

Os arquivos podem ser removidos com `svn remove`:

```
% svn remove foo
```

O subversion não requer a exclusão do arquivo antes de usaramos o comando `svn rm` e, de fato, ele reclama se isso acontecer.

É possível adicionar diretórios com `svn add`:

```
% mkdir bar
% svn add bar
```

Apesar que o `svn mkdir` torna isso mais fácil, combinando a criação do diretório e a adição dele:

```
% svn mkdir bar
```

Como arquivos, os diretórios são removidos com `svn rm`. Não existe um comando separado especificamente para remover diretórios.

```
% svn rm bar
```

5.3.8. Copiando e Movendo Arquivos

Este comando cria uma cópia do arquivo foo.c nomeado bar.c, com o novo arquivo também sob controle de versão e com o histórico completo de foo.c:

```
% svn copy foo.c bar.c
```

Geralmente é preferível copiar desta forma ao invés de copiar o arquivo com comando `cp` e adicionando-o ao repositório com `svn add` porque desta forma o novo arquivo não herda a história do original.

Para mover e renomear um arquivo:

```
% svn move foo.c bar.c
```

5.3.9. Logs e Anotações

O `svn log` mostra revisões e mensagens de commit, as mais recentes são exibidas primeiro, para arquivos ou diretórios. Quando usado em um diretório, todas as revisões que afetaram o diretório e os arquivos nesse diretório são mostradas.

O `svn annotate` , ou igualmente o `svn praise` ou `svn blame`, mostra o número de revisão mais recente e quem fez o commit dessa revisão para cada linha de um arquivo.

5.3.10. Diffs

O `svn diff` exibe alterações na cópia de trabalho. Diffs gerado por `SVN` são unificados e incluem novos arquivos por padrão na saída do diff.

O `svn diff` pode mostrar as mudanças entre duas revisões do mesmo arquivo:

```
% svn diff -r179453:179454 ROADMAP.txt
```

Ele também pode mostrar todas as alterações para um changeset específico. Este comando mostra quais alterações foram feitas no diretório atual e em todos os subdiretórios do changeset 179454:

```
% svn diff -c179454 .
```

5.3.11. Revertendo

Mudanças locais (incluindo adições e deleções) podem ser revertidas usando `svn revert`. Ele não atualiza arquivos desatualizados, apenas os substitui por cópias originais da versão original.

5.3.12. Conflitos

Se um `svn update` resultou em um conflito de mesclagem, o Subversion irá lembrar quais arquivos têm conflitos e se recusará a fazer o commit de quaisquer alterações nesses arquivos até que seja explicitamente informado que os conflitos foram resolvidos. O procedimento simples, ainda não obsoleto, é:

```
% svn resolved foo
```

No entanto, o procedimento preferido é:

```
% svn resolve --accept=working foo
```

Os dois exemplos são equivalentes. Os valores possíveis para `--accept` são:

- `working`: use a versão em seu diretório de trabalho (a qual se presume que foi editada para resolver os conflitos).
- `base`: usar uma cópia original da versão que você tinha antes do `svn update`, descartando suas próprias alterações, as mudanças conflitantes e possivelmente outras mudanças intervenientes também.
- `mine-full`: use o que você tinha antes do `svn update`, incluindo suas próprias mudanças, mas descartando as mudanças conflitantes, e possivelmente outras mudanças intervenientes também.
- `theirs-full`: use a versão que foi recuperada quando você fez o `svn update`, descartando as suas próprias mudanças.

5.4. Uso Avançado

5.4.1. Checkouts dispersos

O SVN permite *sparse*, ou checkouts parciais de um diretório adicionando `--depth` a um `svn checkout`.

Os argumentos válidos para `--depth` são:

- `empty`: o próprio diretório sem qualquer conteúdo.
- `files`: o diretório e quaisquer arquivos nele contidos.
- `immediates`: o diretório e quaisquer arquivos e diretórios contidos nele, mas nenhum dos conteúdos dos subdiretórios.
- `infinity`: qualquer coisa.

A opção `--depth` se aplica a muitos outros comandos, incluindo `svn commit`, `svn revert` e o `svn diff`.

Como o parâmetro `--depth` utilizado é persistente, existe uma opção `--set-depth` para o `svn update` que irá mudar a profundidade selecionada. Assim, dada a cópia de trabalho produzida pelo exemplo anterior:

```
% cd ~/freebsd
% svn update --set-depth=immediates .
```

O comando acima irá popular a cópia de trabalho em *~/freebsd* com ROADMAP.txt e subdiretórios vazios, e nada acontecerá quando `svn update` for executado nos subdiretórios. No entanto, esse comando definirá a profundidade para *head* (nesse caso) como infinito e o populará totalmente:

```
% svn update --set-depth=infinity head
```

5.4.2. Operação Direta

Certas operações podem ser realizadas diretamente no repositório sem tocar na cópia de trabalho. Especificamente, isso se aplica a qualquer operação que não exija a edição de um arquivo, incluindo:

- `log`, `diff`
- `mkdir`
- `remove`, `copy`, `rename`
- `propset`, `propedit`, `propdel`
- `merge`

A criação de uma branch é muito rápida. Este comando seria usado para criar a branch `RELENG_8`:

```
% svn copy svn+ssh://repo.freebsd.org/base/head
```

```
svn+ssh://repo.freebsd.org/base/stable/8
```

Isso equivale a esses comandos, que levam minutos e horas, em vez de segundos, dependendo da sua conexão de rede:

```
% svn checkout --depth=immediates svn+ssh://repo.freebsd.org/base
% cd base
% svn update --set-depth=infinity head
% svn copy head stable/8
% svn commit stable/8
```

5.4.3. Mesclando com SVN

Esta seção lida com o merge de código de um branch para outro (normalmente, do head para um brach stable).



Em todos os exemplos abaixo, `$FSVN` refere-se à localização do repositório Subversion do FreeBSD, `svn+ssh://repo.freebsd.org/base`.

Sobre o acompanhamento de mesclagem

Do ponto de vista do usuário, informações de acompanhamento de mesclagem (ou mergeinfo) são armazenadas em uma propriedade chamada `svn:mergeinfo`, que é uma lista separada por vírgulas de revisões e intervalos de revisões que foram mescladas. Quando definido em um arquivo, ele se aplica somente a esse arquivo. Quando definido em um diretório, ele se aplica a esse diretório e seus descendentes (arquivos e diretórios), exceto aqueles que possuem o seu próprio `svn:mergeinfo`.

Ele *não* é herdado. Por exemplo, stable/6/contrib/openpam/ não herda implicitamente o mergeinfo de stable/6/, ou do stable/6/contrib/. Isso faria com que os checkouts parciais fossem difíceis de gerenciar. Em vez disso, o mergeinfo é explicitamente propagado pela árvore. Para mesclar algo em branch/foo/bar/, estas regras se aplicam:

1. Se branch/foo/bar/ ainda não tiver um registro mergeinfo, mas um ancestral direto (por exemplo, branch/foo/) tem, então esse registro será propagado para abaixo até branch/foo/bar/ antes que as informações sobre a mesclagem atual sejam registradas.
2. As informações sobre a mesclagem atual *não* serão propagadas para o ancestral.
3. Se um descendente direto de branch/foo/bar/ (por exemplo, branch/foo/bar/baz/) já tiver um registro mergeinfo, as informações sobre a mesclagem atual serão propagadas para ele.

Se você considerar o caso em que uma revisão altera várias partes separadas da árvore (por exemplo, branch/foo/bar/ e branch/foo/quux/), mas você só deseja mesclar alguns deles (por exemplo, branch/foo/bar/), você verá que essas regras fazem sentido. Se mergeinfo fosse propagado, pareceria que a revisão também foi mesclada com branch/foo/quux/, quando na verdade não foi.

Selecionando a branch de origem e de destino ao mesclar

Mesclagens para as branches `stable/` devem originar-se da `head/`. Por exemplo:

```
svn merge -c r123456 ^/head/ stable/11
svn commit stable/11
```

Mesclagens para as branches `releng/` devem sempre se originar da branch `stable/` correspondente. Por exemplo:

```
svn merge -c r123456 ^/stable/11 releng/11.0
svn commit releng/11.0
```



Os committers só podem se efetuar um commit para as branches `releng/` durante um ciclo de release após receber aprovação da Equipe de Engenharia de Release, após o qual somente o Security Officer pode efetuar commits para o branch `releng/` para um Aviso de Segurança ou Aviso de Errata.

Todas as mesclagens são mescladas e enviadas por commit a partir da raiz da branch. Todas as mesclagens se parecem com:

```
svn merge -cr123456 ^/head/checkout
svn commit checkout
```

Observe que *checkout* deve ser uma verificação completa da branch na qual a mesclagem ocorre.

```
svn merge -c r123456 ^/stable/10 releng/10.0
```

Preparando o Alvo de Mesclagem (Merge Target)

Devido aos problemas de propagação do mergeinfo descritos anteriormente, é muito importante nunca mesclar as alterações em uma cópia de trabalho esparsa. Sempre use um checkout completo do branch que está sendo mesclado. Por exemplo, ao mesclar de HEAD para 7, use um checkout completo de stable/7:

```
% cd stable/7
% svn up --set-depth=infinity
```

O diretório de destino também deve estar atualizado e não deve conter alterações que ainda não foram enviadas por commit ou arquivos perdidos.

Identificando Revisões

Identificar revisões a serem mescladas é uma obrigação. Se o alvo já tiver mergeinfo completo, solicite uma lista para o `SVN`:

```
% cd stable/6/contrib/openpam
```

```
% svn mergeinfo --show-revs=eligible $FSVN/head/contrib/openpam
```

Se o destino não tiver mergeinfo completo, verifique o log da origem de mesclagem.

Mesclando

Agora vamos começar a mesclar!

Os princípios

Por exemplo, para mesclar:

- revisão $R
- no diretório $target na branch stable $B
- a partir do diretório $source no head
- $FSVN é o svn+ssh://repo.freebsd.org/base

Supondo que as revisões $P e $Q já tenham sido mescladas e que o diretório atual seja uma cópia de trabalho atualizada de stable/$B, a mergeinfo existente será semelhante a:

```
% svn propget svn:mergeinfo -R $target
$target - /head/$source:$P,$Q
```

A mesclagem é feita assim:

```
% svn merge -c$R $FSVN/head/$source $target
```

É possível verificar os resultados disso com um `svn diff`.

O svn:mergeinfo agora se parece com:

```
% svn propget svn:mergeinfo -R $target
$target - head/$source:$P,$Q,$R
```

Se os resultados não forem exatamente como os mostrados, você pode precisar de assistência antes de efetuar o commit pois erros podem ter sido cometidos, ou pode haver algo errado com o mergeinfo existente, ou pode haver um bug no Subversion.

Exemplo Prático

Como um exemplo prático, considere este cenário. As alterações no netmap.4 no r238987 devem ser mescladas do CURRENT para o 9-STABLE. O arquivo reside em head/shared/man/man4. De acordo com o Mesclando com SVN, também é onde devemos fazer a mesclagem. Note que neste exemplo todos os caminhos são relativos ao topo do repositório svn. Para obter mais informações sobre o layout do diretório, consulte Branches RELENG_* e Layout Geral.

O primeiro passo é inspecionar o mergeinfo existente.

```
% svn propget svn:mergeinfo -R stable/9/shared/man/man4
```

Tome uma nota rápida de como ele se parece antes de avançar para o próximo passo; fazendo a mesclagem real:

```
% svn merge -c r238987 svn+ssh://repo.freebsd.org/base/head/shared/man/man4
stable/9/shared/man/man4
--- Merging r238987 into 'stable/9/shared/man/man4':
U    stable/9/shared/man/man4/netmap.4
--- Recording mergeinfo for merge of r238987 into
'stable/9/shared/man/man4':
 U   stable/9/shared/man/man4
```

Verifique se o número de revisão da revisão mesclada foi adicionado. Quando isso for verificado, a única coisa que resta é o commit em si.

```
% svn commit stable/9/shared/man/man4
```

Precauções antes de efetuar o commit

Como sempre, faça um build world (ou as partes apropriadas dele).

Verifique as mudanças com o `svn diff` e `svn stat`. Certifique-se de que todos os arquivos que deveriam ter sido adicionados ou excluídos foram de fato adicionados ou excluídos.

Dê uma olhada mais de perto em qualquer mudança de propriedade (marcada por um `M` na segunda coluna do `svn stat`). Normalmente, nenhuma propriedade svn:mergeinfo deve estar em qualquer lugar, exceto o diretório (ou diretórios) de destino.

Se algo parecer suspeito, peça ajuda.

Fazendo o commit

Certifique-se de efetuar o commit de um diretório de nível superior para incluir também o mergeinfo. Não especifique arquivos individuais na linha de comando. Para mais informações sobre como submeter arquivos em geral, veja a seção relevante deste manual.

5.4.4. Importações de fornecedores com SVN



Por favor, leia toda esta seção antes de iniciar uma importação de fornecedores.



Os patches para o código do fornecedor se enquadram em duas categorias:

- Patches do fornecedor: são patches que foram emitidos pelo fornecedor ou que

foram extraídos do sistema de controle de versão do fornecedor, que abordam problemas que não podem esperar até a próxima versão do fornecedor.

- Patches do FreeBSD: são patches que modificam o código do fornecedor para resolver problemas específicos do FreeBSD.

A natureza de um patch determina para onde ele deve ser enviado por commit:

- Os patches do fornecedor devem ser enviados por commit para o branch do fornecedor e mesclados a partir dele. Se o patch resolver um problema em uma nova versão que está sendo importada atualmente, ele *não deve* ser enviado por commit junto com a nova versão: a versão deve ser importada e marcada primeiro, então a correção pode ser aplicada e o commit efetuado. Não há necessidade de marcar novamente as fontes do fornecedor depois de confirmar o patch.
- Patches do FreeBSD são enviados diretamente para o head.

Preparando a Árvore

Se estiver importando pela primeira vez após a mudança para o Subversion, é necessário achatar e limpar a árvore do fornecedor, bem como inicializar o histórico de mesclagem na árvore principal.

Achatamento

Durante a conversão do `CVS` para o Subversion, as branches do fornecedor foram importadas com o mesmo layout da árvore principal. Isso significa que as fontes do fornecedor `pf` foram armazenadas originalmente em vendor/pf/dist/contrib/pf. O código fonte do fornecedor fica melhor se armazenado diretamente em vendor/pf/dist.

Para achatar a árvore do `pf`:

```
% cd vendor/pf/dist/contrib/pf
% svn mv $(svn list) ../..
% cd ../..
% svn rm contrib
% svn propdel -R svn:mergeinfo .
% svn commit
```

O bit `propdel` é necessário porque, começando com 1.5, o Subversion adicionará `svn:mergeinfo` em qualquer diretório que seja copiado ou movido. Nesse caso, como nada está sendo mesclado da árvore excluída, eles apenas atrapalham.

As tags também podem ser achatadas (3, 4, 3.5 etc.); o procedimento é exatamente o mesmo, mudando apenas `dist` para `3.5` ou similar, e aguardando para executar o `svn commit` apenas no final do processo.

Limpando

A árvore `dist` pode ser limpa conforme necessário. A desativação da expansão de palavras-chave é

recomendada, pois não faz sentido no código do fornecedor não modificado e, em alguns casos, pode até mesmo ser prejudicial. O OpenSSH, por exemplo, inclui dois arquivos que se originaram do FreeBSD e ainda contêm as tags da versão original. Para fazer isso:

```
% svn propdel svn:keywords -R .
% svn commit
```

Bootstrapping o historico de mesclagem

Se estiver importando pela primeira vez após a mudança para o Subversion, faça o bootstrap do `svn:mergeinfo` no diretório de destino da árvore principal para a revisão que corresponde à última alteração relacionada à árvore do fornecedor, antes de importar novos fontes:

```
% cd head/contrib/pf
% svn merge --record-only svn+ssh://repo.freebsd.org/base/vendor/pf/dist@180876 .
% svn commit
```

Importando Novas Fontes

Com dois commits - um para a importação em si e outro para a tag - essa etapa pode ser opcionalmente repetida para cada release upstream entre a última importação e a importação atual.

Preparando os fontes do fornecedor

O Subversion é capaz de armazenar uma distribuição completa na árvore do fornecedor. Portanto, importe tudo, mas mescle apenas o que é necessário.

Um `svn add` é necessário para adicionar quaisquer arquivos que foram adicionados desde a última importação do fornecedor, e o `svn rm` é necessário para remover todos os que foram removidos desde então. Recomenda-se a preparação de listas ordenadas do conteúdo da árvore do fornecedor e das fontes que serão importadas, para facilitar o processo.

```
% cd vendor/pf/dist
% svn list -R | grep -v '/$' | sort >../old
% cd ../pf-4.3
% find . -type f | cut -c 3- | sort >../new
```

Com esses dois arquivos, o `comm -23 ../old ../new` listará os arquivos removidos (arquivos somente em old), enquanto `comm -13 .. /old ../new` listará os arquivos adicionados somente no new.

Importando para a Árvore do Fornecedor

Agora, os fontes devem ser copiados para dist e os comandos `svn add` e `svn rm` são usados conforme necessário:

```
% cd vendor/pf/pf-4.3
```

```
% tar cf - . | tar xf - -C ../dist
% cd ../dist
% comm -23 ../old ../new | xargs svn rm
% comm -13 ../old ../new | xargs svn add --parents
```

Se algum diretório foi removido, ele terá que ser removido manualmente com o `svn rm`. Nada vai quebrar se eles não forem, mas eles permanecerão na árvore.

Verifique as propriedades em qualquer novo arquivo. Todos os arquivos de texto devem ter o `svn:eol-style` definido como `native`. Todos os arquivos binários devem ter o `svn:mime-type` configurado para `application/octet-stream`, a menos que haja um tipo de mídia mais apropriado. Arquivos executáveis devem ter `svn:executable` definido como `*`. Nenhuma outra propriedade deve existir em qualquer arquivo da árvore.

Agora é possível fazer o commit. No entanto, é uma boa prática certificar-se de que tudo está correto, usando os comandos `svn stat` e `svn diff`.

Marcação (Tagging)

Depois de realizado o commit, as versões do fornecedor são marcadas para referência futura. A melhor e mais rápida maneira de fazer isso é diretamente no repositório:

```
% svn cp svn+ssh://repo.freebsd.org/base/vendor/pf/dist
svn+ssh://repo.freebsd.org/base/vendor/pf/4.3
```

Quando isso estiver concluído, execute o `svn up` para a cópia de trabalho do vendor/pf para obter a nova tag, embora isso raramente seja necessário.

Se você criar a tag na cópia de trabalho da árvore, os resultados do `svn:mergeinfo` deverão ser removidos:

```
% cd    vendor/pf
% svn cp dist 4.3
% svn propdel svn:mergeinfo -R 4.3
```

Mesclagem para o Head

```
% cd head/contrib/pf
% svn up
% svn merge --accept=postpone svn+ssh://repo.freebsd.org/base/vendor/pf/dist .
```

O `--accept=postpone` diz ao Subversion para não reclamar sobre conflitos de mesclagem, pois eles serão manipulados manualmente.



A mudança do `cvs2svn` ocorreu em 3 de junho de 2008. Ao realizar mesclagens de fornecedores para pacotes que já estavam presentes e convertidos pelo processo

cvs2svn, o comando usado para mesclar /vendor/package_name/dist para /head/package_location (por exemplo, head/contrib/sendmail) deve usar -c REV para indicar a revisão a ser mesclada a partir da árvore /vendor. Por exemplo:

```
% svn checkout svn+ssh://repo.freebsd.org/base/head/contrib/sendmail
% cd sendmail
% svn merge -c r261190 '^/vendor/sendmail/dist' .
```

^ é um alias para o caminho do repositório.



Se estiver usando o shell Zsh, o ^ deve ser escapado com \ ou entre aspas.

É necessário resolver quaisquer conflitos de mesclagem.

Certifique-se de que todos os arquivos adicionados ou removidos na árvore do fornecedor tenham sido adicionados ou removidos corretamente na árvore principal. Para verificar os diffs com relação ao branch do fornecedor:

```
% svn diff --no-diff-deleted --old=svn+ssh://repo.freebsd.org/base/vendor/pf/dist
--new=.
```

O --no-diff-deleted diz ao Subversion para não reclamar sobre os arquivos que estão na árvore do fornecedor, mas que não estão na árvore principal. Coisas que teriam sido removidas antes da importação do fornecedor, como os makefiles do fornecedor e os scripts de configuração.

Usando o CVS, uma vez que um arquivo estava fora do branch do fornecedor, ele não podia ser colocado de volta. Com o Subversion, não há nenhum conceito dentro ou fora do branch do fornecedor. Se um arquivo que anteriormente tinha modificações locais, para fazer com que ele não apareça em diffs na árvore do fornecedor, tudo o que tem que ser feito é remover qualquer sobra como as tags de versões do FreeBSD, o que é muito mais fácil.

Se alguma mudança for necessária para o "world" compilar com os novos fontes, faça-as agora e continue testando até que tudo seja compilado e executado perfeitamente.

Fazendo o commit da Importação de Fornecedores

Agora é possível fazer o commit! O commit deve ser feito de tudo de uma só vez. Se feito corretamente, a árvore passará de um estado consistente com o código antigo para um estado consistente com o novo código.

A partir do zero

Importando para a Árvore do Fornecedor

Esta seção é um exemplo de importação e marcação do byacc no head.

Primeiro, prepare o diretório em vendor:

```
% svn co --depth immediates $FSVN/vendor
% cd vendor
% svn mkdir byacc
% svn mkdir byacc/dist
```

Agora, importe os fontes para o diretório dist. Uma vez que os arquivos estiverem no lugar, faça o `svn add` dos novos, e então faça o `svn commit` e aplique a tag na versão importada. Para poupar tempo e largura de banda, é possível efetuar diretamente o commit e as marcações de forma remota:

```
% svn cp -m "Tag byacc 20120115" $FSVN/vendor/byacc/dist $FSVN/vendor/byacc/20120115
```

Mesclando para o `head`

Devido a este ser um novo arquivo, copie-o para a mesclagem:

```
% svn cp -m "Import byacc to contrib" $FSVN/vendor/byacc/dist $FSVN/head/contrib/byacc
```

Ainda é possível trabalhar normalmente nos fontes recém importados.

5.4.5. Revertendo um Commit

A reversão de um commit para uma versão anterior é bem fácil:

```
% svn merge -r179454:179453 ROADMAP.txt
% svn commit
```

Alterar a sintaxe do número, com o negativo significando a reversão de uma mudança, também pode ser usado:

```
% svn merge -c -179454 ROADMAP.txt
% svn commit
```

Isso também pode ser feito diretamente no repositório:

```
% svn merge -r179454:179453 svn+ssh://repo.freebsd.org/base/ROADMAP.txt
```



É importante assegurar que o mergeinfo esteja correto ao reverter um arquivo para permitir que o `svn mergeinfo --eligible` funcione como esperado.

A reversão da exclusão de um arquivo é um pouco diferente. É necessário copiar a versão do arquivo que antecede a exclusão. Por exemplo, para restaurar um arquivo que foi excluído na revisão N, restaure a versão N-1:

```
% svn copy svn+ssh://repo.freebsd.org/base/ROADMAP.txt@179454
% svn commit
```

ou, igualmente:

```
% svn copy svn+ssh://repo.freebsd.org/base/ROADMAP.txt@179454
svn+ssh://repo.freebsd.org/base
```

Simplesmente *não* recrie o arquivo manualmente e o adicione com o `svn add` - isso fará com que o histórico seja perdido.

5.4.6. Corrigindo Erros

Por mais que possamos realizar uma cirurgia em uma emergência, não planeje ou corrija erros por baixo dos panos. Planos para erros permanecem nos registros para sempre. Certifique-se de verificar a saída do `svn status` e do `svn diff` antes de enviar o commit.

Erros acontecerão, mas eles geralmente podem ser corrigidos sem perturbar.

No caso de adicionar um arquivo no local errado. A coisa certa a fazer é usar o `svn move` e mover o arquivo para o local correto e fazer o commit. Isso altera apenas algumas linhas de metadados no journal do repositório, e os logs serão todos conectados corretamente.

A coisa errada a fazer é apagar o arquivo e, em seguida, usar `svn add` para adicionar uma cópia independente no local correto. Em vez de algumas linhas de texto, o repositório journal cria uma nova cópia inteira do arquivo. Isso é um desperdício.

5.4.7. Usando um Espelho do Subversion

Há uma séria desvantagem neste método: toda vez que for feito o commit de algo, terá que executar um `svn relocate` no repositório master, não esquecendo de executar o `svn relocate` de volta para o mirror após o commit. Além disso, como o `svn relocate` só funciona entre os repositórios que possuem o mesmo UUID, algum hacking do UUID do repositório local deve ocorrer antes que seja possível começar a usá-lo.

Checkout a partir de um Mirror

Faça o checkout de uma cópia de trabalho a partir de um mirror substituindo a URL do mirror por `svn+ssh://repo.freebsd.org/base`. Este pode ser um mirror oficial ou um mirror mantido usando o `svnsync`.

Configurando um Mirror svnsync

Evite configurar um mirror svnsync a menos que haja uma boa razão para isso. Na maioria das vezes, um mirror `git` é uma alternativa melhor. Começar um novo mirror do zero leva muito tempo. Espere um mínimo de 10 horas para conectividade de alta velocidade. Se o trafego for passar por links internacionais, espere que isso leve de quatro a dez vezes mais.

Uma maneira de limitar o tempo necessário é pegar um arquivo de seed. Ele é grande (~1GB), mas consome menos tráfego de rede e leva menos tempo para baixar do que o svnsync.

Extraia o arquivo e o atualize:

```
% tar xf svnmirror-base-r261170.tar.xz
% svnsync sync file:///home/svnmirror/base
```

Agora, configure isso para executar a partir do cron(8), faça checkouts localmente, configure um servidor svnserve para as máquinas locais com as quais conversar, etc.

O mirror seed está definido para buscar o conteúdo a partir de `svn://svn.freebsd.org/base`. A configuração para o mirror é armazenada em `revprop 0` no mirror local. Para ver a configuração, tente:

```
% svn proplist -v --revprop -r 0 file:///home/svnmirror/base
```

Use `svn propset` para mudar as coisas.

5.4.8. Fazendo commit de dados High ASCII

Os arquivos que possuem bits high-ASCII são considerados arquivos binários no `SVN`, portanto, as verificações de pré-commit falham e indicam que a propriedade `mime-type` deve ser definida como `application/octet-stream`. No entanto, o uso deste é desencorajado, por isso, não o defina. A melhor maneira é sempre evitar dados high-ASCII, para que possam ser lidos em qualquer lugar com qualquer editor de texto, mas se não for possivel evitar, em vez de alterar o mime-type, defina a propriedade `fbsd:notbinary` com o `propset`:

```
% svn propset fbsd:notbinary yes foo.data
```

5.4.9. Mantendo um branch de projeto

Uma branch de projeto é aquela que está sincronizada com o head (ou outra branch) e é usada para desenvolver um projeto e, em seguida, fazer o seu commit de volta para o head. No `SVN`, o branch "dolphin" é usado para isso. Uma branch "dolphin" é aquela que diverge por um tempo e é finalmente enviada de volta ao branch original. Durante a migração do código de desenvolvimento em uma direção (do head para o branch apenas). Não é feito o commit de nenhum código de volta ao head até o final. Depois que é feito o commit da branch no final, ela estará morta (embora uma nova branch com o mesmo nome possa ser criada depois que a que está morta é excluída).

Como explicado em https://people.FreeBSD.org/~peter/svn_notes.txt, o trabalho que destina-se a ser mesclado de volta para o HEAD deve estar em base/projects/. Se o trabalho é benéfico para a comunidade do FreeBSD de alguma forma, mas não se destina a ser mesclado diretamente no HEAD, então o local apropriado é base/user/username/. Esta página contém mais detalhes.

Para criar um branch de projeto:

```
% svn copy svn+ssh://repo.freebsd.org/base/head
svn+ssh://repo.freebsd.org/base/projects/spif
```

Para mesclar as alterações do HEAD de volta ao branch de projeto:

```
% cd copy_of_spif
% svn merge svn+ssh://repo.freebsd.org/base/head
% svn commit
```

É importante resolver quaisquer conflitos de mesclagem antes de fazer o commit.

5.5. Algumas dicas

Nos logs de commit, etc., "rev 179872" é soletrado "r179872" conforme a convenção.

É possível acelerar o svn adicionando estas entradas ao ~/.ssh/config:

```
Host *
ControlPath ~/.ssh/sockets/master-l-r@h:p
ControlMaster auto
ControlPersist yes
```

e depois digitando

```
 mkdir ~/.ssh/sockets
```

Fazer o check-out de uma cópia de trabalho com um cliente Subversion padrão sem patches específicos do FreeBSD (OPTIONS_SET=FREEBSD_TEMPLATE) significará que as tags $FreeBSD: head/pt_BR.ISO8859-1/articles/committers-guide/article.xml 53731 2020-01-01 14:37:53Z ebrandi $ não serão expandidas. Uma vez que a versão correta foi instalada, engane o Subversion para expandi-las da seguinte forma:

```
% svn propdel -R svn:keywords .
% svn revert -R .
```

Isso eliminará os patches para os quais ainda não foi feito o commit.

É possível preencher automaticamente os campos de log de commit "Sponsored by" e "MFC after" configurando os campos "freebsd-sponsored-by" e "freebsd-mfc-after" na seção "[miscellany]" do arquivo de configuração ~/.subversion/config. Por exemplo:

```
freebsd-sponsored-by = The FreeBSD Foundation
freebsd-mfc-after = 2 weeks
```

6. Configuração, Convenções e Tradições

Existe uma série de coisas para fazer como um novo desenvolvedor. O primeiro conjunto de etapas é específico apenas para os committers. Essas etapas devem ser feitas por um mentor para aqueles que não são committers.

6.1. Para novos committers

Aqueles que receberam direitos de commit para os repositórios do FreeBSD devem seguir estes passos.

- Obtenha a aprovação do seu mentor antes de fazer o commit de cada uma dessas mudanças!
- Os arquivos .ent e .xml mencionados abaixo existem no repositório SVN do Projeto de Documentation do FreeBSD em `svn+ssh://repo.FreeBSD.org/doc/`.
- Novos arquivos que não possuem a propriedade `FreeBSD=%H svn:keywords` serão rejeitados quando você tentar fazer o commit dos mesmos para o repositório. Certifique-se de ler Adicionando e Removendo Arquivos sobre como adicionar e remover arquivos. Verifique se o ~/.subversion/config contém as entradas necessárias de "auto-props" do auto-props.txt mencionado lá.
- Todos os commits do src vão para o FreeBSD-CURRENT antes de serem mesclados para o FreeBSD-STABLE. A branch do FreeBSD-STABLE deve manter a compatibilidade de ABI e de API com as versões anteriores dessa branch. Não mescle as alterações que quebram essa compatibilidade.

Procedure: Etapas para os novos committers

1. Adicione uma entidade de autor

   doc/head/shared/xml/authors.ent - Adicione uma entidade de autor. Etapas posteriores dependem dessa entidade, e a falta dessa etapa fará com que a compilação do doc/ falhe. Essa é uma tarefa relativamente fácil, mas continua sendo um bom primeiro teste de habilidades no controle de versão.

2. Atualize a lista de desenvolvedores e colaboradores

   doc/head/en_US.ISO8859-1/articles/contributors/contrib.committers.xml - Adicione uma entrada à seção "Desenvolvedores" da Lista de Contribuidores. As entradas são classificadas pelo sobrenome.

   doc/head/en_US.ISO8859-1/articles/contributors/contrib.additional.xml - *Remova* a entrada da seção "Contribuidores Adicionais". As entradas são classificadas pelo primeiro nome.

3. Adicione um item de notícias

   doc/head/shared/xml/news.xml - Adicione uma entrada. Procure por outras entradas que anunciam novos committers e siga o formato. Use a data do email de aprovação do core@FreeBSD.org para o commit bit.

4. Adicione uma chave PGP

   doc/head/shared/pgpkeys/pgpkeys.ent e doc/head/shared/pgpkeys/pgpkeys-developers.xml - Adicione a sua chave PGP ou GnuPG. Aqueles que ainda não têm uma chave devem ver Criando uma chave.

   O Dag-Erling Smørgrav des@FreeBSD.org escreveu um script de shell (doc/head/shared/pgpkeys/addkey.sh) para tornar isto mais fácil. Consulte o arquivo README para obter mais informações.

   Use o doc/head/shared/pgpkeys/checkkey.sh para verificar se as chaves atendem aos padrões mínimos das boas práticas recomendadas.

   Depois de adicionar e verificar uma chave, adicione os dois arquivos atualizados ao controle de versão do código fonte, e em seguida faça o commit. As entradas neste arquivo são classificadas pelo sobrenome.

   

   É muito importante ter uma chave PGP/GnuPG atualizada no repositório. A chave pode ser necessária para a identificação positiva de um committer. Por exemplo, os Administradores do FreeBSD admins@FreeBSD.orgpodem precisar dela para a recuperação da conta. Um chaveiro completo dos usuários do `FreeBSD.org` está disponível para download em https://www.FreeBSD.org/doc/pgpkeys.txt.

5. Atualize as informações do Mentor e do Mentee

   base/head/shared/misc/committers-repository.dot - Adicione uma entrada à seção committers atuais, onde *repository* é `doc`, `ports`, ou `src`, dependendo dos privilégios de commit concedidos.

   Adicione uma entrada para cada relacionamento adicional de mentor/mentee na seção inferior.

6. Gere uma senha do Kerberos

   Veja Kerberos e LDAP Web Password para o cluster do FreeBSD para gerar ou definir um Kerberos para uso com outros serviços do FreeBSD, como o banco de dados de rastreamento de bugs.

7. Opcional: Ative a sua conta na Wiki

   Conta no Wiki do FreeBSD - Uma conta no wiki permite que compartilhe projetos e ideias. Aqueles que ainda não têm uma conta podem seguir as instruções da Página Sobre a Wiki para obter uma. Entre em contato com wikiadmin@FreeBSD.org se precisar de ajuda com sua conta no Wiki.

8. Opcional: Atualize informações do Wiki

   Informações do Wiki - Depois de obter acesso ao wiki, algumas pessoas adicionam entradas nas páginas How We Got Here, IRC Nicks e Dogs of FreeBSD páginas.

9. Opcional: Atualize o Ports com as suas Informações Pessoais

ports/astro/xearth/files/freebsd.committers.markers e src/usr.bin/calendar/calendars/calendar.freebsd - Algumas pessoas adicionam entradas para elas nestes arquivos para mostrar onde estão localizadas ou a data de seu aniversário.

10. Opcional: Evite Correspondências Duplicadas

    Assinantes das listas svn-src-all, svn-ports-all ou da svn-doc-all podem desejar cancelar a assinatura para evitar receber cópias duplicadas de mensagens de commit e de followups.

6.2. Para todos

1. Apresente-se aos outros desenvolvedores, caso contrário, ninguém fará a menor ideia de quem você é ou no que você está trabalhando. A introdução não precisa ser uma biografia abrangente, basta escrever um parágrafo ou dois sobre quem você é, em que você planeja trabalhar como desenvolvedor no FreeBSD e quem será seu mentor. Envie os parágrafos por e-mail para a lista de discussão dos desenvolvedores do FreeBSD e você estará a caminho!
2. Entre na `freefall.FreeBSD.org` e crie um arquivo /var/forward/user (no qual *user* é seu nome de usuário) contendo o endereço de e-mail para o qual você deseja que o correio endereçado para *yourusername*@FreeBSD.org seja encaminhado. Isto inclui todas as mensagens de commit assim como qualquer outro email endereçado à lista de discussão dos committers do FreeBSD e à lista de discussão dos desenvolvedores do FreeBSD. Caixas de correio realmente grandes que tenham residência permanente na `freefall` podem ser truncadas sem aviso se o espaço precisar ser liberado, então encaminhe suas mensagens ou salve-as em outro lugar.

   

   Se o seu sistema de e-mail usa SPF com regras estritas, você deve colocar o host `mx2.FreeBSD.org` na whitelist de verificações do SPF.

   Devido ao load severo imposto aos servidores centrais que fazem o processamento da lista de discussão devido ao grande volume de SPAMs, o servidor de front-end faz algumas verificações básicas e descarta algumas mensagens com base nessas verificações. No momento, as informações de DNS adequadas para o host de conexão são a única verificação ativa, mas isso pode mudar. Algumas pessoas culpam estas verificações por rejeitar e-mails válidos. Para que essas verificações sejam desativadas no seu email, crie um arquivo chamado ~/.spam_lover no servidor `freefall.FreeBSD.org`.



Aqueles que são desenvolvedores, mas não são committers, não serão inscritos nas listas de discussão de desenvolvedores ou de committers. As assinaturas são derivadas dos direitos de acesso.

6.2.1. Configuração de acesso SMTP

Para aqueles dispostos a enviar mensagens de e-mail através da infraestrutura do FreeBSD.org, siga as instruções abaixo:

1. Aponte seu cliente de e-mail para `smtp.FreeBSD.org:587`.
2. Habilite o STARTTLS.
3. Assegure-se de que seu endereço `From:` esteja configurado para `yourusername@FreeBSD.org`.

4. Para autenticação, você pode usar o seu nome de usuário e a sua senha do Kerberos no cluster do FreeBSD (veja Kerberos e LDAP Web Password para o cluster do FreeBSD). O principal `yourusername/mail` é o preferido, pois é válido apenas para autenticação dos recursos de correio.



Não inclua `@FreeBSD.org` ao digitar seu nome de usuário.

- Aceitará apenas mensagens de `yourusername@FreeBSD.org`. Se você for autenticado como um usuário, não terá permissão para enviar e-mails como outro.
- Um cabeçalho será anexado com o nome de usuário do SASL: (`Authenticated sender: username`).
- O host possui vários limites de taxa para reduzir as tentativas de força bruta.

Usando um MTA Local para Encaminhar Emails para o Serviço SMTP do FreeBSD.org

Também é possível usar um MTA local para encaminhar emails enviados localmente para os servidores SMTP do FreeBSD.org.

Exemplo 1. Usando o Postfix

Para dizer a uma instância local do Postfix que qualquer email de `yourusername@FreeBSD.org` deve ser encaminhado para os servidores do FreeBSD.org, adicione isto ao seu main.cf:

```
sender_dependent_relayhost_maps = hash:/usr/local/etc/postfix/relayhost_maps
smtp_sasl_auth_enable = yes
smtp_sasl_security_options = noanonymous
smtp_sasl_password_maps = hash:/usr/local/etc/postfix/sasl_passwd
smtp_use_tls = yes
```

Crie /usr/local/etc/postfix/relayhost_maps com o seguinte conteúdo:

```
yourusername@FreeBSD.org  [smtp.freebsd.org]:587
```

Crie /usr/local/etc/postfix/sasl_passwd com o seguinte conteúdo:

```
[smtp.freebsd.org]:587          yourusername:yourpassword
```

Se o servidor de email for usado por outras pessoas, talvez você queira impedir que elas enviem emails do seu endereço. Para configurar isso, adicione estas entradas ao seu main.cf:

```
smtpd_sender_login_maps = hash:/usr/local/etc/postfix/sender_login_maps
smtpd_sender_restrictions = reject_known_sender_login_mismatch
```

Crie /usr/local/etc/postfix/sender_login_maps com o seguinte conteúdo:

```
yourusername@FreeBSD.org yourlocalusername
```

Onde *yourlocalusername* é o usuário SASL utilizado para conectar na instância local do Postfix.

6.3. Mentores

Todos os novos desenvolvedores têm um mentor atribuído a eles nos primeiros meses. Um mentor é responsável por ensinar ao aprendiz as regras e convenções do projeto e orientar seus primeiros passos na comunidade de desenvolvedores. O mentor também é pessoalmente responsável pelas ações do aprendiz durante este período inicial.

Para committers: não faça nenhum commit sem antes obter a aprovação do seu mentor. Documente essa aprovação com uma linha `Approved by:` na mensagem de commit.

Quando o mentor decide que um aprendiz já aprendeu o necessário e está pronto para fazer commits por conta própria, o mentor anuncia o fato com um commit no conf/mentors. Este arquivo está na branch svnadmin de cada repositório:

`src`	base/svnadmin/conf/mentors
`doc`	doc/svnadmin/conf/mentors
`ports`	ports/svnadmin/conf/mentors



Os novos committers devem ter como objetivo realizar commits o suficiente para que seu mentor se sinta confortável em liberá-los da mentoria no primeiro ano. Se eles ainda estiverem sob orientação, o corpo gerencial apropriado (core, doceng ou portmgr) deve tentar garantir que não haja barreiras que impeçam a conclusão. Se o committer não for capaz de satisfazer seu mentor de prontidão por um ano e meio, seu commit bit pode ser convertido para membro do projeto.

7. Revisão pré-commit

A revisão de código é uma maneira de aumentar a qualidade do software. As seguintes diretrizes se aplicam a commits na ramificação `head` (-CURRENT) do repositório `src`. Outras ramificações e as árvores do `ports` e do `docs` têm suas próprias políticas de revisão, mas essas diretrizes geralmente se aplicam a commits que exigem revisão:

- Todas as alterações não triviais devem ser revisadas antes de serem cometidas no repositório.
- As revisões podem ser conduzidas por e-mail, pelo Bugzilla, pelo Phabricator ou por outro mecanismo. Sempre que possível, as revisões devem ser públicas.
- O desenvolvedor responsável por uma mudança de código também é responsável por fazer todas as alterações necessárias relacionadas à revisão.
- A revisão de código pode ser um processo iterativo, que continua até que o patch esteja pronto para o commit. Especificamente, uma vez que um patch é enviado para revisão, ele deve

receber um explícito "looks good" antes que o commit possa ser feito. Desde que a aprovação seja explícita, ela pode ser formalizada de qualquer forma que faça sentido para o método de revisão.

- Timeouts não são um substituto para revisão.

Às vezes, as revisões de código demoram mais tempo do que você esperaria, especialmente para recursos maiores. As formas aceitas de acelerar os tempos de revisão dos seus patches são:

- Revise os patches de outras pessoas. Se você ajudar, todos estarão mais dispostos a fazer o mesmo por você; A boa vontade é a nossa moeda.
- Ping o patch. Se for urgente, forneça as razões pelas quais é importante para você finalizá-lo e faça o ping a cada dois dias. Se não for urgente, a frequência adequada de ping é de uma vez por semana. Lembre-se de que você está pedindo um tempo valioso de outros desenvolvedores profissionais.
- Peça ajuda em listas de discussão, IRC, etc. Outros podem ajudá-lo diretamente ou sugerir um revisor.
- Dividir seu patch em vários patches pequenos os quais se aplicam uns sobre os outros. Quanto menor o seu patch, maior a probabilidade de alguém dar uma rápida olhada nele.

  Ao fazer grandes mudanças, é útil ter isso em mente desde o início do esforço, pois quebrar grandes alterações em pequenas é geralmente difícil depois que estão prontas.

Os desenvolvedores devem participar de revisões de código fazendo revisões e recebendo revisões. Se alguém tiver a gentileza de revisar seu código, você deve devolver o favor a outra pessoa. Observe que, embora qualquer pessoa seja bem-vinda para revisar e dar feedback sobre um patch, apenas um especialista no assunto apropriado pode aprovar uma alteração. Geralmente, este especialista será um committer que trabalha com o código em questão regularmente.

Em alguns casos, nenhum especialista no assunto pode estar disponível. Nesses casos, uma revisão por um desenvolvedor experiente é suficiente quando associada a testes apropriados.

8. Mensagens de Log de Commit

Esta seção contém algumas sugestões e tradições de como os logs de commit são formatados.

Além de incluir uma mensagem informativa com cada commit, algumas informações adicionais podem ser necessárias.

Essas informações consistem em uma ou mais linhas contendo a palavra-chave ou frase, dois-pontos, guias para formatação e, em seguida, as informações adicionais.

As palavras ou frases-chave são:

`PR:`	O relatório do problema (se houver) que é afetado (normalmente, por estar fechado) por este commit. Vários PRs podem ser especificados em uma linha, separados por vírgulas ou espaços.

`Submitted by:`	O nome e o endereço de e-mail da pessoa que enviou a correção; para desenvolvedores, apenas o nome de usuário no cluster do FreeBSD. Se o requisitante for o mantenedor do port que está sendo atualizado pelo commit, inclua "(maintainer)" após o endereço de e-mail. Evite ofuscar o endereço de e-mail do remetente, pois isso adiciona trabalho adicional ao pesquisar os registros.
`Reviewed by:`	O nome e o endereço de e-mail da pessoa ou pessoas que revisaram a alteração; para desenvolvedores, apenas o nome de usuário no cluster do FreeBSD. Se um patch foi submetido a uma lista de discussão para revisão e a revisão foi favorável, basta incluir o nome da lista.

`Approved by:`	O nome e endereço de e-mail da pessoa ou pessoas que aprovaram a alteração; para desenvolvedores, apenas o nome de usuário no cluster do FreeBSD. É costume obter aprovação prévia para um commit se for para uma área da árvore na qual você normalmente não faz commit. Além disso, durante a preparação para uma nova versão, todos os commits *devem* ser aprovados pela equipe de engenharia de release.
	
	Enquanto estiver sob orientação, obtenha aprovação do mentor antes do commit. Digite o nome de usuário do mentor neste campo e adicione uma nota de que ele é um mentor:
	
	`Approved by: username-of-mentor (mentor)`
	
	Se uma equipe aprovou esses commits, inclua o nome da equipe seguido do nome de usuário do aprovador entre parênteses. Por exemplo:
	
	`Approved by: re (username)`

`Obtained from:`	O nome do projeto (se houver) do qual o código foi obtido. Não use esta linha para o nome de uma pessoa individual.
`Sponsored by:`	Organizações patrocinadoras dessa mudança, se houver. Separe várias organizações com vírgulas. Se apenas uma parte do trabalho foi patrocinada, ou diferentes montantes de patrocínio foram fornecidos a diferentes autores, por favor, forneça os devidos créditos entre parênteses após cada nome de patrocinador. Por exemplo, `Example.com (alice, refatoração de código), Wormulon (bob), Momcorp (cindy)` mostra que Alice foi patrocinada pela Example.com para refatoração de código, enquanto Wormulon patrocinou o trabalho de Bob e a Momcorp patrocinou o trabalho de Cindy. Outros autores não foram patrocinados ou optaram por não listar o patrocínio.
`MFC after:`	Para receber um lembrete por e-mail para o MFC em uma data posterior, especifique o número de dias, semanas ou meses após os quais um MFC está planejado.

`MFC to:`	Se o commit deve ser mesclado em um subconjunto de branches estáveis, especifique os nomes das branchs.
`MFC with:`	Se o commit deve ser mesclado junto com um commit anterior em um único MFC (por exemplo, onde o commit corrige um bug da alteração anterior), especifique o número de revisão correspondente.
`Relnotes:`	Se a alteração for candidata a inclusão nas notas de lançamento da próxima versão da branch, defina como `yes`.
`Security:`	Se a alteração estiver relacionada a uma vulnerabilidade de segurança ou exposição de segurança, inclua uma ou mais referências ou uma descrição do problema. Se possível, inclua um URL VuXML ou um ID CVE.
`Evento:`	Descrição para o evento em que essa confirmação foi feita. Se este for um evento recorrente, adicione o ano ou até mesmo o mês. Por exemplo, isso pode ser `FooBSDcon 2019`. A idéia por trás dessa linha é dar reconhecimento a conferências, reuniões e outros tipos de reuniões e mostrar que elas são úteis. Por favor, não use a linha `Patrocinado por:` para isso, pois isso significa que as organizações patrocinam determinados recursos ou desenvolvedores que trabalham neles.
`Differential Revision:`	A URL completa da revisão do Phabricator. Esta linha *deve ser a última linha* . Por exemplo: `https://reviews.freebsd.org/D1708`.

Exemplo 2. Commit Log para um commit baseado em um PR

O commit é baseado em um patch de um PR enviado por John Smith. Os campos da mensagem de commit "PR" e "Submitted by" são preenchidos.

```
...

        PR:                     12345
        Submitted by:       John Smith <John.Smith@example.com>
```

Exemplo 3. Commit log de um commit que precisa de revisão

O sistema de memória virtual está sendo alterado. Depois de enviar os patches para a lista de discussão apropriada (neste caso, `freebsd-arch`) e as mudanças foram aprovadas.

```
...
        Reviewed by:        -arch
```

Exemplo 4. Commit log de um commit que precisa de aprovação

Commit um port, depois de trabalhar com o MAINTAINER, que disse para ir em frente e executar o commit.

```
...
        Approved by:        abc (maintainer)
```

No qual o *abc* é o nome da conta da pessoa que aprovou.

Exemplo 5. Commit log de um commit que importa código do OpenBSD

Efetuando o commit de códigos baseados no trabalho feito no projeto OpenBSD.

```
...
        Obtained from:      OpenBSD
```

Exemplo 6. Commit Log para uma mudança no FreeBSD-CURRENT com um commit planejado para o FreeBSD-STABLE para seguir em uma data posterior.

Efetuando o commit de códigos que serão mesclados do FreeBSD-CURRENT na branch do FreeBSD-STABLE após duas semanas.

```
...
MFC after:      2 weeks
```

Onde *2* é o número de dias, semanas ou meses após o qual um MFC é planejado. A opção *weeks* pode ser `day`, `dias`, `semana`, `semanas`, `mês`, `meses`.

Muitas vezes é necessário combinar isso.

Considere a situação em que um usuário enviou um código contendo o PR do projeto NetBSD. Olhando para o PR, o desenvolvedor vê que não é uma área da árvore na qual eles normalmente trabalham, então eles têm a mudança revisada pela lista de discussão `arch`. Como a mudança é complexa, o desenvolvedor opta pelo MFC após um mês para permitir testes adequados.

A informação extra para incluir no commit seria algo como

Exemplo 7. Exemplo de log de commit combinado

```
PR:                 54321
Submitted by:       John Smith <John.Smith@example.com>
Reviewed by:        -arch
Obtained from:      NetBSD
MFC after:          1 month
Relnotes:           yes
```

9. Licença preferida para novos arquivos

A política de licença completa do Projeto FreeBSD pode ser encontrada em https://www.FreeBSD.org/internal/software-license/. O restante desta seção destina-se a ajudá-lo a começar. Por via de regra, quando em dúvida, pergunte. É muito mais fácil dar conselhos do que consertar a árvore de código fonte.

O Projeto FreeBSD sugere e usa este texto como o esquema de licença preferencial:

```
/*-
 * SPDX-License-Identifier: BSD-2-Clause-FreeBSD
 *
 * Copyright (c) [year] [your name]
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form 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.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``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 AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * [id for your version control system, if any]
 */
```

O projeto FreeBSD desencoraja fortemente a chamada "cláusula publicitária" no novo código. Devido ao grande número de contribuidores para o projeto FreeBSD, o cumprimento desta cláusula para muitos fornecedores comerciais se tornou difícil. Se você tiver um código na árvore com a cláusula de publicidade, considere removê-lo. Na verdade, considere usar a licença acima para o seu código.

O projeto FreeBSD desencoraja completamente novas licenças e variações nas licenças padrões. Novas licenças requerem a aprovação do Core Team core@FreeBSD.org para residir no repositório principal. Quanto mais licenças diferentes forem usadas na árvore, mais problemas isso causará para aqueles que desejarem utilizar esse código, geralmente de consequências não intencionais de

uma licença mal formulada.

A política do projeto determina que o código sob algumas licenças não-BSD deve ser colocado apenas em seções específicas do repositório e, em alguns casos, a compilação deve ser condicional ou até mesmo desativada por padrão. Por exemplo, o kernel GENERIC deve ser compilado apenas sob licenças idênticas ou substancialmente semelhantes à licença BSD. O software licenciado GPL, APSL, CDDL, etc, não deve ser compilado no GENERIC.

Os desenvolvedores são lembrados de que, em código aberto, ficar "aberto" corretamente é tão importante quanto obter o "fonte" correto, já que o manuseio inadequado da propriedade intelectual tem sérias consequências. Quaisquer dúvidas ou preocupações devem imediatamente ser levadas à atenção do Core Team.

10. Acompanhando as Licenças Concedidas ao Projeto FreeBSD

Vários softwares ou dados existem nos repositórios onde o projeto FreeBSD recebeu uma licença especial para poder usá-los. Um caso em questão são as fontes do Terminus para uso com o vt(4). Aqui, o autor Dimitar Zhekov nos permitiu usar a fonte "Terminus BSD Console" sob uma licença BSD de 2 cláusulas, em vez da licença Open Font License que ele normalmente usa.

É claramente sensível manter um registro de tais concessões de licença. Para esse fim, o Core Team core@FreeBSD.org decidiu manter um arquivo delas. Sempre que o projeto FreeBSD receber uma licença especial, nós exigimos que o Core Team core@FreeBSD.org seja notificado. Qualquer desenvolvedor envolvido na obtenção de tal concessão de licença, por favor envie os detalhes para o Core Team core@FreeBSD.org incluindo:

- Detalhes de contato para as pessoas ou organizações que concederam a licença especial.
- Quais arquivos, diretórios etc. nos repositórios são cobertos pela concessão da licença, incluindo os números de revisão dos commits nos quais qualquer material especialmente licenciado tenha sido incorporado.
- A data em que a licença entra em vigor. Salvo acordo em contrário, esta será a data em que a licença foi emitida pelos autores do software em questão.
- O texto da licença.
- Uma nota de quaisquer restrições, limitações ou exceções que se aplicam especificamente ao uso do material licenciado pelo FreeBSD.
- Qualquer outra informação relevante.

Uma vez que o Core Team core@FreeBSD.org esteja satisfeito de que todos os detalhes necessários foram reunidos e estão corretos, o secretário enviará um aviso de recebimento assinado com o PGP, incluindo os detalhes da licença. Esse recibo será persistentemente arquivado e servirá como nosso registro permanente da concessão da licença.

O arquivo de licença deve conter apenas detalhes das concessões de licença; este não é o lugar para qualquer discussão em torno de licenciamento ou outros assuntos. O acesso aos dados dentro do arquivo de licença estará disponível mediante solicitação para o Core Team core@FreeBSD.org.

11. Relações entre os Desenvolvedores

Ao trabalhar diretamente em seu próprio código ou em um código que já esteja bem estabelecido como sua responsabilidade, provavelmente haverá pouca necessidade de verificar com outros committers antes de entrar com um commit. Ao trabalhar em um bug em uma área do sistema que é claramente órfã (e existem algumas dessas áreas, para nossa vergonha), o mesmo se aplica. Ao modificar partes do sistema que são mantidas, formalmente ou informalmente, considere solicitar uma revisão exatamente como um desenvolvedor teria de fazer antes de se tornar um committer. Para os ports, entre em contato com o `MAINTAINER` listado no Makefile.

Para determinar se uma área da árvore é mantida, verifique o arquivo MAINTAINERS na raiz da árvore. Se ninguém estiver listado, analise o histórico de revisões para ver quem fez o commit de alterações no passado. Um script de exemplo que lista cada pessoa que já fez commit de um determinado arquivo junto com o número de commits que cada pessoa fez pode ser encontrado na `freefall` em ~eadler/bin/whodid. Se as consultas não forem atendidas ou se o committer indicar uma falta de interesse na área afetada, vá em frente e faça o commit.



Evite enviar e-mails privados para os mantenedores. Outras pessoas podem estar interessadas na conversa, não apenas no resultado final.

Se houver qualquer dúvida sobre um commit por qualquer razão, submeta-o ao processo de revisão antes de faze-lo. É melhor fazer com que ele seja criticado lá e não quando ele já fizer parte do repositório. Se um commit resulta em controvérsia, pode ser aconselhável considerar a possibilidade de fazer o rollback do commit até que o assunto seja resolvido. Lembre-se, com um sistema de controle de versão, podemos sempre alterá-lo de volta.

Não impugne as intenções dos outros. Se eles vêem uma solução diferente para um problema, ou até um problema diferente, provavelmente não é porque eles são estúpidos, porque eles têm parentesco questionável, ou porque eles estão tentando destruir o trabalho duro, a imagem pessoal ou o FreeBSD, mas basicamente porque eles têm uma visão diferente do mundo. Diferente é bom.

Discorde honestamente. Argumente sua posição com base em seus méritos, seja honesto quanto a quaisquer deficiências que possa ter, e esteja aberto para ver a solução deles, ou mesmo a visão deles do problema, com uma mente aberta.

Aceite a correção. Somos todos falíveis. Quando você cometer um erro, peça desculpas e continue com a vida. Não se bata, e certamente não espanque os outros por seu erro. Não perca tempo com constrangimentos ou recriminações, apenas conserte o problema e siga em frente.

Peça por ajuda. Procure (e dê) revisões dos seus pares. Uma das maneiras pelas quais o software de código aberto deve se sobressair é no número de globos oculares aplicados a ele; isso não se aplica se ninguém revisar o código.

12. Se em dúvida...

Quando não tiver certeza sobre algo, seja um problema técnico ou uma convenção do projeto, não se esqueça de perguntar. Se você ficar em silêncio, nunca fará progressos.

Se estiver relacionado a um problema técnico, pergunte nas listas públicas de discussão. Evite a tentação de enviar e-mail para a pessoa que conhece a resposta. Dessa forma, todos poderão aprender com a pergunta e a resposta.

Para perguntas específicas ou administrativas do projeto, pergunte na seguinte ordem:

- Seu mentor ou ex-mentor.
- Um committer experiente no IRC, email, etc.
- Qualquer equipe com um "hat" (chapéu), uma vez que eles podem lhe dar uma resposta definitiva.
- Se ainda não tiver certeza, pergunte na lista de discussão dos desenvolvedores do FreeBSD.

Uma vez que sua pergunta seja respondida, se ninguém lhe indicou a documentação que soletra a resposta para sua pergunta, documente-a, pois outros terão a mesma pergunta no futuro.

13. Bugzilla

O Projeto FreeBSD utiliza o Bugzilla para rastrear bugs e requisições de alteração. Certifique-se de fechar o PR caso você faça o commit de uma correção ou sugestão encontrada no banco de dados de PR. Também é considerada uma boa prática você reservar um tempo para fechar qualquer PR associado aos seus commits, se apropriado.

Committers com contas não-`FreeBSD.org` no Bugzilla podem ter a conta antiga mesclada com a conta `FreeBSD.org` seguindo estes passos:

1. Faça o login usando sua conta antiga.
2. Abra um novo bug. Escolha `Services` como o Produto e `Bug Tracker` como o Componente. Na descrição de erros liste as contas que você deseja mesclar.
3. Faça o login usando a conta do `FreeBSD.org` e poste um comentário no bug recém-aberto para confirmar a propriedade. Veja Kerberos e LDAP Web Password para o cluster do FreeBSD para mais detalhes sobre como gerar ou definir uma senha para sua conta `FreeBSD.org`.
4. Se houver mais de duas contas para mesclar, poste comentários de cada uma delas.

Você pode descobrir mais sobre o Bugzilla em:

* * https://www.FreeBSD.org/support/

14. Phabricator

O projeto FreeBSD utiliza o Phabricator para solicitações de revisão de código. Veja a página CodeReview no wiki para detalhes.

Committers com contas não-`FreeBSD.org` no Phabricator podem ter a conta antiga renomeada para a conta `FreeBSD.org` seguindo estes passos:

1. Altere o email da conta do Phabricator para o seu email `FreeBSD.org`.

2. Abra um novo bug em nosso bug tracker usando sua conta `FreeBSD.org`, veja Bugzilla para mais informações. Escolha `Services` como o Produto e `Code Review` como o Componente. Na descrição de bugs, solicite que a sua conta do Phabricator seja renomeada e forneça um link para o usuário do Phabricator. Por exemplo, `https://reviews.freebsd.org/p/bob_example.com/`



As contas do Phabricator não podem ser mescladas, por favor, não abra uma nova conta.

15. Quem é quem

Além dos repositórios meisters, existem outros membros e equipes do projeto FreeBSD que você provavelmente conhecerá no exercício da sua função como committer. Resumidamente, e de forma alguma inclusivamente, estes são:

Equipe de Engenharia de Documentação doceng@FreeBSD.org

O doceng é o grupo responsável pela infraestrutura de criação de documentação, aprovando de novos committers de documentação e garantindo que o website do FreeBSD e a documentação no site FTP estão atualizados em relação à árvore subversion. Não é um corpo de resolução de conflitos. A grande maioria das discussões relacionadas à documentação ocorre na lista de discussão do projeto de documentação do FreeBSD. Mais detalhes sobre a equipe doceng podem ser encontrados em seu charter. Os committers interessados em contribuir com a documentação devem se familiarizar com o Primer do Projeto de Documentação.

Glen Barber gjb@FreeBSD.org, Konstantin Belousov kib@FreeBSD.org, Bryan Drewery bdrewery@FreeBSD.org, Marc Fonvieille blackend@FreeBSD.org, Xin Li delphij@FreeBSD.org, Colin Percival cperciva@FreeBSD.org Hiroki Sato hrs@FreeBSD.org, Gleb Smirnoff glebius@FreeBSD.org

Estes são os membros da Equipe de Engenharia de Release re@FreeBSD.org. Essa equipe é responsável por definir os prazos de lançamento e por controlar o processo de release. Durante o congelamento de código, os engenheiros de release têm autoridade final sobre todas as alterações no sistema para qualquer branch que esteja com status de release pendente. Se há algo que você deseja mesclar do FreeBSD-CURRENT para o FreeBSD-STABLE (quaisquer valores que eles possam ter em um dado momento), estas são as pessoas com quem conversar sobre isso.

Gordon Tetlow gordon@FreeBSD.org

Gordon Tetlow é o Oficial de Segurança do FreeBSD e supervisiona a Equipe Oficial de Segurança security-officer@FreeBSD.org.

Garrett Wollman wollman@FreeBSD.org

Se você precisar de conselhos sobre aspectos internos obscuros da rede ou não tiver certeza de alguma mudança potencial no subsistema de rede que você tem em mente, Garrett é alguém com quem conversar. Garrett também é muito conhecedor dos vários padrões aplicáveis ao FreeBSD.

Lista de discussão dos committers do FreeBSD

svn-src-all, svn-ports-all e svn-doc-all são as listas de discussão que o sistema de controle de

versão usa para enviar mensagens de commit. *Nunca* envie e-mail diretamente para essas listas. Apenas envie respostas para esta lista quando elas forem curtas e estiverem diretamente relacionadas a um commit.

Lista de discussão dos desenvolvedores do FreeBSD

Todos os committers estão inscritos na -developers. Esta lista foi criada para ser um fórum para os problemas da "comunidade" dos committers. Exemplos são a eleição do Core Team, anúncios, etc.

A lista de discussão dos desenvolvedores do FreeBSD é para uso exclusivo dos committers do FreeBSD. Para desenvolver o FreeBSD, os committers devem ter a capacidade de discutir abertamente assuntos que serão resolvidos antes de serem anunciados publicamente. As discussões francas sobre o trabalho em andamento não são adequadas para publicação aberta e podem prejudicar o FreeBSD.

Espera-se que todos os committers do FreeBSD não publiquem ou encaminhem mensagens da lista de discussão dos desenvolvedores do FreeBSD fora da lista de membros sem a permissão de todos os autores. Violadores serão removidos da lista de discussão dos desenvolvedores do FreeBSD, resultando em uma suspensão dos privilégios de commit. Violações repetidas ou flagrantes podem resultar na revogação permanente dos privilégios de commit.

Não é a intenção desta lista ser um local para revisões de código ou para realizar qualquer discussão técnica. Na verdade, usá-lo como tal prejudica o Projeto FreeBSD, pois dá uma sensação de uma lista fechada, onde as decisões gerais que afetam toda a comunidade usando o FreeBSD são feitas sem serem "abertas". Por último, mas não menos importante, *nunca, nunca, envie por e-mail a lista de discussão dos desenvolvedores do FreeBSD e faça CC:/BCC: para outra lista do FreeBSD*. Nunca, jamais envie email para outra lista de emails do FreeBSD e faça CC:/BCC: para a lista de discussão dos desenvolvedores do FreeBSD. Fazer isso pode diminuir muito os benefícios dessa lista.

16. Guia de início rápido do SSH

1. Se você não quiser digitar sua senha toda vez que usar o ssh(1), e você usa chaves para autenticar, o ssh-agent(1) está lá para sua conveniência. Se você quiser usar o ssh-agent(1), certifique-se de executá-lo antes de executar os outros aplicativos. Os usuários de X, por exemplo, geralmente fazem isso a partir do .xsession ou do .xinitrc. Veja ssh-agent(1) para detalhes.
2. Gere um par de chaves usando ssh-keygen(1). O par de chaves será colocado no diretório $HOME/.ssh/.

   

   Somente as chaves ECDSA, Ed25519 ou RSA são suportadas.

3. Envie sua chave pública ($HOME/.ssh/id_ecdsa.pub, $HOME/.ssh/id_ed25519.pub, ou $HOME/.ssh/id_rsa.pub) para a pessoa que está configurando você como um committer para que ela possa ser colocada em yourlogin in /etc/ssh-keys/ na `freefall`. Agora ssh-add(1) pode ser usado para autenticação uma vez por sessão. Ele solicita a frase secreta da chave privada e a armazena no agente de autenticação (ssh-agent(1)). Use o `ssh-add -d` para remover as chaves

armazenadas no agente.

Teste com um simples comando remoto: `ssh freefall.FreeBSD.org ls /usr`.

Para obter mais informações, consulte security/openssh-portable, ssh(1), ssh-add(1), ssh-agent(1), ssh-keygen(1), e scp(1).

Para obter informações sobre como adicionar, alterar ou remover chaves ssh(1), consulte este artigo.

17. Disponibilidade do Coverity® para os Committers do FreeBSD

Todos os desenvolvedores do FreeBSD podem obter acesso aos resultados da análise do Coverity de todo o software do Projeto FreeBSD. Todos os interessados em obter acesso aos resultados de análise das execuções automatizadas do Coverity podem se inscrever em Coverity Scan.

O wiki do FreeBSD inclui um mini-guia para desenvolvedores interessados em trabalhar com os relatórios de análise do Coverity®: https://wiki.freebsd.org/CoverityPrevent. Por favor observe que este mini-guia só pode ser lido por desenvolvedores do FreeBSD, então se você não puder acessar esta página, você terá que pedir a alguém para adicioná-lo à lista de acesso apropriada do Wiki.

Finalmente, todos os desenvolvedores do FreeBSD que usarão o Coverity® são sempre encorajados a pedir mais detalhes e informações sobre o uso, publicando quaisquer perguntas na lista de discussão dos desenvolvedores do FreeBSD.

18. A Grande Lista de Regras dos Committers do FreeBSD

Todos os envolvidos com o projeto FreeBSD devem obedecer ao *Código de Conduta* disponível em https://www.FreeBSD.org/internal/code-of-conduct/. Como committers, vocês formam a face pública do projeto, e como você se comporta tem um impacto vital na percepção pública disso. Este guia expande as partes do *Código de Conduta* específico para os committers.

1. Respeite outros committers.
2. Respeite os outros colaboradores.
3. Discuta qualquer mudança significativa *antes de* fazer o commit.
4. Respeite os mantenedores existentes (se listado no campo `MAINTAINER` no Makefile ou no MAINTAINER no diretório de nível superior).
5. Deve ser feito o rollback de qualquer alteração contestada enquanto estiver pendente a resolução da disputa, se solicitado por um mantenedor. Alterações relacionadas à segurança podem anular os desejos de um mantenedor, a critério do Oficial de Segurança.
6. As alterações vão para o FreeBSD-CURRENT antes do FreeBSD-STABLE, a menos que especificamente permitido pelo engenheiro de release ou a menos que elas não sejam aplicáveis

ao FreeBSD-CURRENT. Qualquer alteração não trivial ou não urgente que seja aplicável também deve ser permitida no FreeBSD-CURRENT por pelo menos 3 dias antes da fusão, para que tenha tempo suficiente de teste. O engenheiro de release tem a mesma autoridade sobre o branch FreeBSD-STABLE, conforme descrito para o mantenedor na regra #5.

7. Não brigue em público com outros committers; parece ruim.
8. Respeite todos os congelamentos de código e leia as listas de discussão `committers` e `developers` em tempo hábil para que você saiba quando um congelamento de código está em vigor.
9. Em caso de dúvida em qualquer procedimento, pergunte primeiro!
10. Teste suas alterações antes de fazer o commit.
11. Não commit em software contribuído sem a aprovação *explicita* dos respectivos mantenedores.

Conforme observado, a quebra de algumas dessas regras pode ser motivo para suspensão ou, mediante ofensa repetida, remoção permanente de privilégios de commit. Membros individuais do core têm o poder de suspender temporariamente os privilégios de commit até que o core como um todo tenha a chance de revisar o problema. No caso de uma "emergência" (um committer causando dano ao repositório), uma suspensão temporária também pode ser feita pelos meisters do repositório. Apenas uma maioria de 2/3 do core tem autoridade para suspender os privilégios de confirmação por mais de uma semana ou para removê-los permanentemente. Essa regra não existe para definir o core como um bando de ditadores cruéis que podem se livrar casualmente de committers como se fossem latas de refrigerante vazias, mas para dar ao projeto uma espécie de fusível de segurança. Se alguém está fora de controle, é importante ser capaz de lidar com isso imediatamente, em vez de ficar paralisado pelo debate. Em todos os casos, um committer cujos privilégios foram suspensos ou revogados tem direito a uma "audiência" com o core, sendo a duração total da suspensão determinada naquele momento. Um committer cujos privilégios são suspensos também pode solicitar uma revisão da decisão após 30 dias e a cada 30 dias a partir de então (a menos que o período total de suspensão seja inferior a 30 dias). Um committer cujos privilégios tenham sido revogados completamente pode solicitar uma revisão após um período de 6 meses. Esta política de revisão é *estritamente informal* e, em todos os casos, o core reserva-se o direito de agir ou desconsiderar os pedidos de revisão se eles sentirem que a decisão original é a correta.

Em todos os outros aspectos da operação do projeto, o core é um subconjunto de committers e é limitado pelas *mesmas regras*. Só porque alguém está no core isso não significa que eles têm permissão especial para sair de qualquer uma das linhas pintadas aqui; os "poderes especiais" do core só são aplicados quando ele age como um grupo, não individualmente. Como indivíduos, os membros da equipe principal são todos committers em primeiro lugar e core em segundo.

18.1. Detalhes

1. Respeite outros committers.

   Isso significa que você precisa tratar outros committers como os desenvolvedores de grupos pares que eles são. Apesar de nossas tentativas ocasionais de provar o contrário, não se chega a ser um committer por ser estúpido e nada incomoda mais do que ser tratado dessa maneira por um de seus colegas. Se nós sempre sentimos respeito uns pelos outros ou não (e todo mundo tem dias difíceis), nós ainda temos que *tratar* os outros committers com respeito em todos os

momentos, em fóruns públicos e em emails privados.

Ser capaz de trabalhar juntos a longo prazo é o maior patrimônio deste projeto, muito mais importante do que qualquer conjunto de alterações no código, e transformar argumentos sobre código em problemas que afetam nossa capacidade de longo prazo de trabalhar harmoniosamente juntos não vale a pena por qualquer estiramento concebível da imaginação.

Para cumprir esta regra, não envie e-mails quando estiver com raiva ou de alguma forma se comportar de uma maneira que possa causar uma confrontação desnecessária com os outros. Primeiro acalme-se, então pense em como se comunicar da maneira mais eficaz para convencer as outras pessoas de que o seu lado do argumento está correto, não apenas gaste um pouco de vapor para que você possa se sentir melhor a curto prazo às custas de uma guerra de longa duração. Isso não só é uma "economia de energia" muito ruim, mas demonstrações repetidas de agressão pública que prejudicam nossa capacidade de trabalhar bem juntas serão tratadas severamente pela liderança do projeto e podem resultar na suspensão ou término dos seus privilégios de commit. . A liderança do projeto levará em consideração as comunicações públicas e privadas trazidas a ela. Ela não buscará a divulgação de comunicações privadas, mas levará isso em conta se for oferecida de forma voluntária pelos envolvidos na reclamação.

Tudo isso nunca é uma opção que a liderança do projeto goste nem um pouco, mas a união vem em primeiro lugar. Nenhuma quantidade de código ou de bons conselhos vale a pena se trocar desta forma.

2. Respeite os outros colaboradores.

   Você nem sempre foi um committer. Houve uma época em que você era um colaborador. Lembre-se disso em todos os momentos. Lembre-se de como foi tentar obter ajuda e atenção. Não se esqueça de que seu trabalho como colaborador foi muito importante para você. Lembre-se de como foi. Não desencoraje, deprecie ou diminua os colaboradores. Trate-os com respeito. Eles são nossos committers em espera. Eles são tão importantes para o projeto quanto os committers. Suas contribuições são tão válidas e tão importantes quanto as suas. Afinal, você fez muitas contribuições antes de se tornar um committer. Sempre se lembre disso.

   Considere os pontos levantados sob Respeite outros committers e aplique-os também aos contribuidores.

3. Discuta qualquer mudança significativa *antes de* fazer o commit.

   O repositório não é onde as alterações são inicialmente submetidas para correção ou discussão, isso acontece primeiro nas listas de discussão ou pelo uso do serviço do Phabricator. O commit só acontecerá quando algo semelhante a um consenso for alcançado. Isso não significa que a permissão seja necessária antes de corrigir todos os erros óbvios de sintaxe ou erros ortográficos na página manual, significa apenas que é bom desenvolver uma ideia de quando a mudança proposta não é tão óbvia e requer algum feedback primeiro. As pessoas realmente não se importam com mudanças radicais se o resultado for algo claramente melhor do que antes, elas simplesmente não gostam de ser *surpreendidas* por essas mudanças. A melhor maneira de certificar-se de que as coisas estão no caminho certo é ter o código revisado por um ou mais committers.

   Em caso de dúvida, peça por uma revisão!

4. Respeite os mantenedores existentes, se listados.

   Muitas partes do FreeBSD não são "possuídas" no sentido de que qualquer indivíduo específico irá pular e gritar se você enviar uma alteração para a "sua" área, mas ainda vale a pena verificar primeiro. Uma convenção que usamos é colocar uma linha de mantenedor no Makefile para qualquer pacote ou subárvore que esteja sendo mantido ativamente por uma ou mais pessoas; veja Source Tree Guidelines and Policies para documentação sobre isso. Nas seções de código para quais existirem vários mantenedores, os commits nas áreas afetadas por um mantenedor precisarão ser revisados por pelo menos um outro mantenedor. Nos casos em que o "maintainer-ship" de algo não está claro, consulte os logs do repositório para os arquivos em questão e veja se alguém está trabalhando recentemente ou predominantemente naquela área.

5. Deve ser feito o rollback de qualquer alteração contestada enquanto estiver pendente a resolução da disputa, se solicitado por um mantenedor. Alterações relacionadas à segurança podem anular os desejos de um mantenedor, a critério do Oficial de Segurança.

   Isso pode ser difícil de engolir em momentos de conflito (quando cada lado está convencido de que eles estão certos, é claro), mas um sistema de controle de versão torna desnecessário ter uma disputa em andamento quando é muito mais fácil simplesmente reverter a mudança que gerou a disputa, faça com que todos se acalmem novamente e tente descobrir qual é a melhor maneira de proceder. Se a mudança acaba por ser a melhor coisa depois de tudo, ela pode ser facilmente trazida de volta. Se ela não for, os usuários não terão que viver com a mudança falsa na árvore enquanto todos estavam ocupados debatendo seus méritos. Pessoas *muito* raramente pedem rollbacks no repositório, uma vez que a discussão geralmente expõe mudanças ruins ou controversas antes que o commit aconteça, mas em raras ocasiões o rollback deve ser feito sem discussão para que possamos entrar imediatamente da discussão do tópico para descobrirmos se ele era adequado ou não.

6. As alterações vão para o FreeBSD-CURRENT antes do FreeBSD-STABLE, a menos que especificamente permitido pelo engenheiro de release ou a menos que elas não sejam aplicáveis ao FreeBSD-CURRENT. Qualquer alteração não trivial ou não urgente que seja aplicável também deve ser permitida no FreeBSD-CURRENT por pelo menos 3 dias antes da fusão, para que possa ter tempo suficiente de teste. O engenheiro de lançamento tem a mesma autoridade sobre o branch FreeBSD-STABLE, conforme descrito na regra #5.

   Este é outro problema do tipo "não discuta sobre isso", já que é o engenheiro de release quem é o responsável final (e é espancado) se uma mudança for ruim. Por favor, respeite isso e dê ao engenheiro de release a sua total cooperação quando se trata do branch FreeBSD-STABLE. O gerenciamento do FreeBSD-STABLE pode frequentemente parecer excessivamente conservador para o observador casual, mas também deve ter em mente o fato de que o conservadorismo deve ser a marca do FreeBSD-STABLE e regras diferentes aplicam-se lá do que no FreeBSD-CURRENT. Também não há sentido em fazer com que o FreeBSD-CURRENT seja um campo de testes se as alterações forem mescladas no FreeBSD-STABLE imediatamente. Mudanças precisam de uma chance de serem testadas pelos desenvolvedores do FreeBSD-CURRENT, então espere algum tempo antes da fusão, a menos que a correção do FreeBSD-STABLE seja crítica, sensível ao tempo ou óbvia a ponto de tornar desnecessário testes adicionais (correções ortográficas nas páginas de manual) correções de erros / erros de digitação, etc.) Em outras palavras, aplique o bom senso.

Mudanças nas branches de segurança (por exemplo, `releng/9.3`) devem ser aprovadas por um membro da Equipe de Segurança security-officer@FreeBSD.org, ou em alguns casos , por um membro da Equipe de Engenharia de Release re@FreeBSD.org.

7. Não brigue em público com outros committers; parece ruim.

   Este projeto tem uma imagem pública a defender e essa imagem é muito importante para todos nós, especialmente se quisermos continuar a atrair novos membros. Haverá ocasiões em que, apesar das melhores tentativas de autocontrole de todos, os ânimos se perdem e palavras de raiva são trocadas. A melhor coisa que pode ser feita nesses casos é minimizar os efeitos disso até que todos tenham esfriado. Não divulgue palavras iradas em público e não encaminhe correspondências privadas ou outras comunicações privadas para listas publicas de discussão, aliases de mensagens, canais de mensagens instantâneas ou sites de mídia social. O que as pessoas dizem um-para-um é frequentemente muito menos revestido de açúcar do que o que eles diriam em público, e tais comunicações, portanto, não têm lugar lá - elas servem apenas para inflamar uma situação já ruim. Se a pessoa que enviou um flame-o-grama tiver pelo menos a elegância de enviá-lo em particular, então tenha a elegância de mantê-lo em sigilo. Se você acha que está sendo tratado injustamente por outro desenvolvedor e está lhe causando angústia, traga o assunto para o core em vez de torná-lo público. O Core fará o seu melhor para atuar como pacificadores e trazer as coisas de volta para a sanidade. Nos casos em que a disputa envolve uma alteração na base de código e os participantes não parecem estar chegando a um acordo amigável, o core pode nomear um terceiro mutuamente aceitável para resolver a disputa. Todas as partes envolvidas devem então concordar em se comprometer com a decisão tomada por este terceiro.

8. Respeite todos os congelamentos de código e leia atempadamente a lista de discussão `committers` e `developers` para saber quando um congelamento de código está em vigor.

   Efetuar o commit de alterações não aprovadas durante um congelamento de código é um erro realmente grande e espera-se que os committers se mantenham atualizados sobre o que está acontecendo antes de entrar depois de uma longa ausência e fazer o commit de 10 megabytes de material acumulado. As pessoas que abusarem disso regularmente terão seus privilégios de commit suspensos até que eles voltem do FreeBSD Happy Reeducation Camp que mantemos na Groenlândia.

9. Em caso de dúvida em qualquer procedimento, pergunte primeiro!

   Muitos erros são cometidos porque alguém está com pressa e apenas assume que sabe a forma certa para fazer alguma coisa. Se você não fez isso antes, é bem provável que você não conheça realmente a maneira como fazemos as coisas e realmente precise perguntar primeiro ou você vai se envergonhar completamente em público. Não há vergonha em perguntar "como diabos eu faço isso?" Já sabemos que você é uma pessoa inteligente; caso contrário, você não seria um committer.

10. Teste suas alterações antes de fazer o commit.

    Isso pode parecer óbvio, mas se realmente fosse tão óbvio, provavelmente não veríamos tantos casos de pessoas claramente não fazendo isso. Se suas mudanças são para o kernel, certifique-se de que você ainda pode compilar o GENERIC e o LINT. Se as suas alterações estiverem em outro

lugar, certifique-se de que você ainda pode fazer um "make world". Se as alterações forem feitas em uma branch, certifique-se de que seu teste ocorra com uma máquina que esteja executando esse código. Se você tiver uma alteração que também possa quebrar outra arquitetura, verifique e teste em todas as arquiteturas suportadas. Por favor, consulte a Página Interna do FreeBSD para obter uma lista dos recursos disponíveis. À medida que outras arquiteturas são adicionadas à lista de plataformas suportadas do FreeBSD, os recursos de teste compartilhados apropriados serão disponibilizados.

11. Não commit em software contribuído sem a aprovação *explicita* dos respectivos mantenedores.

    Software contribuído é qualquer código que esteja sob as árvores src/contrib, src/crypto, ou src/sys/contrib.

    As árvores mencionadas acima são para software contribuído geralmente importado para um branch de fornecedor. Fazer o commit de algo lá pode causar dores de cabeça desnecessárias quando for importado novas versões do software. Uma regra geral é considerar enviar os patches upstream diretamente para o fornecedor. Patches podem ser committados primeiramente no FreeBSD, desde que tenha a permissão do mantenedor.

    As razões para modificar o software upstream variam entre querer controle estrito sobre uma dependência fortemente acoplada à falta de portabilidade na distribuição do repositório canônico do seu código. Independentemente do motivo, o esforço para minimizar a carga de manutenção do fork é útil para outros mantenedores. Evite realizar commits de alterações triviais ou cosméticas nos arquivos, pois isso dificulta cada merge: esses patches precisam ser verificados manualmente a cada importação.

    Se uma parte específica do software não tiver um mantenedor, você é incentivado a assumir a propriedade. Se você não tiver certeza sobre o mantenedor atual, envie um email para a lista de email de arquitetura e design do FreeBSD e pergunte.

18.2. Política sobre Várias Arquiteturas

O FreeBSD adicionou ports para várias novas arquiteturas durante os ciclos de release recentes e realmente não é mais um sistema operacional centrado em i386™. Em um esforço para tornar mais fácil manter o FreeBSD portátil entre as plataformas que suportamos, o core desenvolveu este mandato:

Nossa plataforma de referência de 32 bits é a i386 e a nossa plataforma de referência de 64 bits é amd64. O principal trabalho de design (incluindo as principais alterações da API e da ABI) deve ser comprovado em pelo menos uma plataforma de 32 bits e pelo menos uma de 64 bits, preferencialmente nas plataformas de referência primária, antes que o seu commit possa ser feito na árvore de fontes.

As plataformas i386 e amd64 foram escolhidas por estarem mais prontamente disponíveis para os desenvolvedores e como representantes dos mais diversos designs de processador e de sistema - "big versus little endian", registrador de arquivos versus pilha de registro, diferentes implementações de DMA e cache, tabelas de página de hardware versus gerenciamento de TLB de software, etc.

Continuaremos a reavaliar essa política, já que o custo e a disponibilidade das plataformas de 64 bits mudam.

Os desenvolvedores também devem estar cientes da nossa Política de Tier para o suporte de longo prazo das arquiteturas de hardware. As regras aqui pretendem fornecer orientação durante o processo de desenvolvimento e são diferentes dos requisitos de recursos e arquiteturas listados nessa seção. As regras de Tier para suporte de recursos em arquiteturas no momento da release são mais rigorosas do que as regras para alterações durante o processo de desenvolvimento.

18.3. Outras Sugestões

Ao enviar alterações na documentação, use um verificador ortográfico antes de efetuar o commit. Para todos os documentos XML, verifique se as diretivas de formatação estão corretas executando o `make lint` e o textproc/igor.

Para as páginas de manual, execute o sysutils/manck e o textproc/igor na página de manual para verificar se todas as referências cruzadas e referências de arquivo estão corretas e se a página man possui todos os `MLINKS` apropriados instalados.

Não misture correções de estilo com novas funcionalidades. Uma correção de estilo é qualquer alteração que não modifique a funcionalidade do código. Misturar as alterações ofusca a mudança de funcionalidade ao solicitar a comparação das diferenças entre as revisões, o que pode ocultar quaisquer novos bugs. Não inclua alterações de espaço em branco com alterações de conteúdo nos commits para doc/. A desordem extra nos diffs torna o trabalho dos tradutores muito mais difícil. Em vez disso, faça qualquer alteração de estilo ou espaço em branco em commits separados e que sejam claramente rotuladas como tal na mensagem de commit.

18.4. Recursos Obsoletos (Deprecated)

Quando for necessário remover uma funcionalidade de software do sistema básico, siga estas diretrizes sempre que possível:

1. Uma menção é feita na página de manual e, possivelmente, nas notas de versão que a opção, o utilitário ou a interface estão obsoletos. O uso do recurso obsoleto gera um aviso.
2. A opção, utilitário ou interface é preservada até a próxima release principal (ponto zero).
3. A opção, o utilitário ou a interface são removidos e não são mais documentados. Agora está obsoleto. Também é geralmente uma boa ideia anotar sua remoção nas notas de release.

18.5. Privacidade e Confidencialidade

1. A maioria dos negócios do FreeBSD é feita em público.

   O FreeBSD é um projeto *aberto*. O que significa que não só alguém pode usar o código-fonte, mas que a maior parte do processo de desenvolvimento está aberto ao escrutínio público.

2. Certos assuntos delicados devem permanecer privados ou mantidos sob embargo.

   Infelizmente, não pode haver transparência completa. Como desenvolvedor do FreeBSD, você

terá um certo grau de acesso privilegiado à informação. Consequentemente, espera-se que você respeite certos requisitos de confidencialidade. Às vezes, a necessidade de confidencialidade vem de colaboradores externos ou tem um limite de tempo específico. Principalmente, porém, é uma questão de não liberar comunicações privadas.

3. O oficial de segurança tem controle exclusivo sobre a liberação de alertas de segurança.

   Onde existem problemas de segurança que afetam muitos sistemas operacionais diferentes, o FreeBSD frequentemente depende do acesso antecipado para poder preparar alertas para liberação coordenada. A menos que se possa confiar nos desenvolvedores do FreeBSD para manter a segurança, esse acesso antecipado não será disponibilizado. O Oficial de Segurança é responsável por controlar o acesso pré-lançamento às informações sobre vulnerabilidades, e por definir o momento de liberação de todos os alertas. Ele pode solicitar ajuda sob condição de confidencialidade de qualquer desenvolvedor com conhecimento relevante para preparar as correções de segurança.

4. As comunicações com o Core são mantidas confidenciais pelo tempo que for necessário.

   As comunicações para o core serão inicialmente tratadas como confidenciais. Eventualmente, no entanto, a maioria dos negócios do Core serão resumidos nos relatórios mensais ou trimestrais do core. Cuidado será tomado para evitar a divulgação de detalhes sensíveis. Os registros de alguns assuntos particularmente sensíveis podem não ser relatados e serão mantidos apenas nos arquivos privados do Core.

5. Acordos de não divulgação (NDA) podem ser necessários para o acesso a determinados dados comercialmente sensíveis.

   O acesso a determinados dados comercialmente sensíveis só pode estar disponível sob um Contrato de Não Divulgação. A equipe jurídica da Fundação FreeBSD deve ser consultado antes de qualquer acordo vinculativo ser firmado.

6. Comunicações privadas não devem ser tornadas públicas sem permissão.

   Além dos requisitos específicos acima, há uma expectativa geral de não publicar comunicações privadas entre desenvolvedores sem o consentimento de todas as partes envolvidas. Peça permissão antes de encaminhar uma mensagem para uma lista de discussão pública, ou publicá-la em um fórum ou site que possa ser acessado por outros que não os correspondentes originais.

7. As comunicações em canais de acesso restrito ou somente do projeto devem ser mantidas em sigilo.

   Semelhantemente às comunicações pessoais, certos canais de comunicação interna, incluindo as listas de discussão dos Committers do FreeBSD e os canais de IRC de acesso restrito, são considerados comunicações privadas. A permissão é necessária para publicar material dessas fontes.

8. O Core pode aprovar a publicação.

   Quando for impraticável obter permissão devido ao número de correspondentes ou quando a

permissão para publicar for injustificadamente retida, a Core poderá aprovar a liberação de tais assuntos privados que mereçam uma publicação mais geral.

19. Suporte para Várias Arquiteturas

O FreeBSD é um sistema operacional altamente portátil destinado a funcionar em muitos tipos diferentes de arquiteturas de hardware. Manter uma separação clara entre o código dependente de máquina (MD) e independente de máquina (MI), além de minimizar o código MD, é uma parte importante da nossa estratégia de permanecer ágil em relação às tendências atuais de hardware. Cada nova arquitetura de hardware suportada pelo FreeBSD aumenta substancialmente o custo de manutenção de código, de suporte do toolchain e de engenharia de release. Também aumenta drasticamente o custo do teste efetivo das alterações no kernel. Como tal, há uma forte motivação para diferenciar as classes de suporte para as várias arquiteturas, permanecendo forte em algumas arquiteturas chaves que são vistas como o "público-alvo" do FreeBSD.

19.1. Declaração de Intenção Geral

O projeto FreeBSD tem como objetivo a "qualidade comercial de produção off-the-shelf (COTS) para workstations, servidores e sistemas embarcados de ponta". Mantendo o foco em um conjunto restrito de arquiteturas de interesse nesses ambientes, o Projeto FreeBSD é capaz de manter altos níveis de qualidade, estabilidade e desempenho, bem como de minimizar a carga de várias equipes de suporte no projeto, como a equipe de ports, a equipe de documentação, o oficial de segurança e as equipes de engenharia de release. A diversidade no suporte de hardware amplia as opções para os consumidores do FreeBSD oferecendo novos recursos e oportunidades de uso, mas esses benefícios devem sempre ser cuidadosamente considerados em termos do custo de manutenção no mundo real associado ao suporte adicional à plataforma.

O projeto FreeBSD diferencia plataforma alvos em quatro camadas. Cada camada inclui uma lista de garantias que os consumidores podem confiar , bem como as obrigações por Projeto e desenvolvedores para cumprir essas garantias. Esta lista define o mínimo de garantia por cada camada. O Projeto e desenvolvedores podem prover leveis de suporte adicionais além da garantia mínima dada a uma camada, mas tais suportes adicionais não tem garantias. Cada plataforma alvo é designada para uma camada especifica para cada ramificação stable. Como resultado, a plataforma alvo poderia ser designada para diferentes camadas em ramificações stable concorrentes.

19.2. Plataformas Alvo

O suporte para uma plataforma de hardware consiste em dois componentes: suporte ao kernel e interfaces binárias de aplicativos (ABIs) do usuário. O suporte do kernel à plataforma inclui os itens necessários para executar um kernel do FreeBSD em uma plataforma de hardware, como gerenciamento de memória virtual dependente de máquina e drivers de dispositivo. Uma ABI especifica uma interface para os processos do usuário interagirem com o kernel do FreeBSD e as bibliotecas do sistema base. Uma ABI inclui interfaces de chamada do sistema, o layout e a semântica de estruturas de dados públicos e o layout e a semântica de argumentos transmitidos às sub-rotinas. Alguns componentes de uma ABI podem ser definidos por especificações tais como o layout dos objetos de exceção C ++ ou as convenções de chamada para funções C.

Um kernel do FreeBSD também usa uma ABI (às vezes chamada de Interface Binária do Kernel (KBI)) que inclui a semântica e layouts de estruturas de dados públicos e o layout e semântica de argumentos para funções públicas dentro do próprio kernel.

Um kernel do FreeBSD pode suportar múltiplas ABIs de usuário. Por exemplo, o kernel amd64 do FreeBSD suporta as ABIs de usuário do FreeBSD amd64 e i386, bem como as ABIs de usuário do Linux x86_64 e i386. Um kernel do FreeBSD deve suportar uma ABI "nativa" como a ABI padrão. A "ABI" nativa geralmente compartilha certas propriedades com a ABI do kernel, como a convenção de chamada C, tamanhos dos tipos básicos, etc.

Os Tiers são definidos tanto para os kernels quanto para as ABIs do usuário. Normalmente, o kernel de uma plataforma e as ABIs do FreeBSD são atribuídas à mesma Tier.

19.3. Tier 1: Arquiteturas Completamente Suportadas

As plataformas Tier 1 são as plataformas mais maduras do FreeBSD. Elas são suportadas pelas equipes de segurança, engenharia de release e gerenciamento de ports. Espera-se que as arquiteturas Tier 1 estejam com Qualidade de Produção em relação a todos os aspectos do sistema operacional FreeBSD, incluindo ambientes de instalação e desenvolvimento.

O Projeto FreeBSD fornece as seguintes garantias aos consumidores das plataformas de Tier 1:

- Imagens oficiais de Release do FreeBSD serão fornecidas pela equipe de engenharia de release.
- Serão fornecidas atualizações binárias e patches no formato de código fonte para os Avisos de Segurança e Avisos de Erro das versões suportadas.
- Serão fornecidos patches de código fonte para os avisos de segurança das branches suportadas.
- Atualizações binárias e patches de código fonte geralmente são fornecidos para os avisos de segurança multiplataforma no momento do anúncio.
- As alterações nas ABIs do usuário geralmente incluirão bibliotecas de compatibilidade para garantir a operação correta dos binários compilados a partir de qualquer branch estável de uma plataforma Tier 1. Estas bibliotecas podem não ser ativadas na instalação padrão. Se as bibliotecas de compatibilidade não forem fornecidas para uma alteração de ABI, a falta da mesma estará claramente documentada nas notas de versão.
- Alterações em certas partes da ABI do kernel incluirão bibliotecas de compatibilidade para garantir a operação correta dos módulos do kernel compilados contra a versão suportada mais antiga da branch. Observe que nem todas as partes da ABI do kernel estão protegidas.
- Pacotes binários oficiais para softwares de terceiros serão fornecidos pela equipe de ports. Para arquiteturas embarcadas, esses pacotes podem ser compilados de forma cruzada a partir de uma arquitetura diferente.
- Os ports mais relevantes devem ou compilar ou ter filtros apropriados para prevenir a compilação dos inadequados.
- Novos recursos que não são inerentemente específicos da plataforma serão totalmente funcionais em todas as arquiteturas de Tier 1.
- Os recursos e bibliotecas de compatibilidade usados pelos binários compilados a partir das branchs estáveis mais antigas podem ser removidos nas versões principais mais recentes. Essas

remoções serão claramente documentadas nas notas de versão.

- As plataformas Tier 1 devem ser totalmente documentadas. As operações básicas serão documentadas no Handbook do FreeBSD.
- As plataformas de Tier 1 serão incluídas na árvore de código fonte.
- As plataformas de Tier 1 devem ser auto-hospedadas ou por meio de um toolchain disponível na árvore de código fonte ou por meio de um toolchain externo. Se um toolchain externo for necessário, serão fornecidos pacotes binários oficiais para o toolchain externo.

Para manter a maturidade das plataformas de Tier 1, o Projeto FreeBSD manterá os seguintes recursos para apoiar o desenvolvimento:

- Suporte à automação do processo de compilação e de testes no cluster FreeBSD.org ou em algum outro local facilmente disponível para todos os desenvolvedores. Plataformas embarcadas podem substituir um emulador disponível no cluster FreeBSD.org por hardware real.
- A inclusão nos targets do `make universe` e `make tinderbox`.
- Hardware dedicado em um dos clusters do FreeBSD para compilação de pacotes (nativamente ou via qemu-user).

Coletivamente, os desenvolvedores devem fornecer o seguinte para manter o status de Tier 1 de uma plataforma:

- Alterações na árvore de código fonte não devem quebrar conscientemente a compilação de uma plataforma de Tier 1.
- As arquiteturas de Tier 1 devem ter um ecossistema maduro e saudável de usuários e desenvolvedores ativos.
- Os desenvolvedores devem ser capazes de compilar pacotes em sistemas Tier 1 não-embarcados comumente disponíveis. Isso pode significar compilações nativas se sistemas não-embarcados estiverem normalmente disponíveis para a plataforma em questão, ou significar compilações cruzadas hospedadas em alguma outra arquitetura de Tier 1.
- As alterações não podem quebrar a ABI do usuário. Se for necessária uma alteração da ABI, a compatibilidade da ABI com os binários existentes deve ser provida através do versionamento de símbolos ou do versionamento de bibliotecas compartilhadas.
- Alterações mescladas em branchs estáveis não podem quebrar as partes protegidas da ABI do kernel. Se for necessária uma alteração da ABI do kernel, ela deverá ser modificada para preservar a funcionalidade dos módulos existentes do kernel.

19.4. Tier 2: Arquiteturas de Desenvolvimento e Nicho

As plataformas de Tier 2 são funcionais, mas são plataformas FreeBSD menos maduras. Elas não são suportadas pelas equipes de segurança, engenharia de release e gerenciamento de ports.

As plataformas Tier 2 podem ser candidatas à plataforma Tier 1 que ainda estão em desenvolvimento ativo. As arquiteturas que atingem o fim da vida útil também podem ser movidas do status de Tier 1 para o status de Tier 2 à medida que a disponibilidade de recursos para continuar a manter o sistema em um estado de Qualidade de Produção diminui. Arquiteturas de

nicho bem suportadas também podem ser de Tier 2.

O Projeto FreeBSD fornece as seguintes garantias aos consumidores das plataformas de Tier 2:

- A infraestrutura de ports deve incluir suporte básico para as arquiteturas de Tier 2 suficiente para suportar a compilação de ports e pacotes. Isso inclui suporte para pacotes básicos, tais como o ports-mgmt/pkg, mas não há garantia de que ports arbitrários serão compiláveis ou funcionais.
- Novos recursos que não são inerentemente específicos da plataforma devem ser viáveis em todas as arquiteturas de Tier 2 se não implementados.
- As plataformas de Tier 2 serão incluídas na árvore de código fonte.
- As plataformas de Tier 2 devem ser auto-hospedadas ou por meio de um toolchain disponível na árvore de código fonte ou por meio de um toolchain externo. Se um toolchain externo for necessário, serão fornecidos pacotes binários oficiais para o toolchain externo.
- As plataformas de Tier 2 devem fornecer kernels e espaço de usuário funcionais, mesmo que uma release oficial não seja provida.

Para manter a maturidade das plataformas de Tier 2, o Projeto FreeBSD manterá os seguintes recursos para apoiar o desenvolvimento:

- A inclusão nos targets do `make universe` e `make tinderbox`.

Coletivamente, os desenvolvedores devem fornecer o seguinte para manter o status de Tier 2 de uma plataforma:

- Alterações na árvore de código fonte não devem quebrar conscientemente a compilação de uma plataforma de Tier 2.
- As arquiteturas de Tier 2 devem ter um ecossistema ativo de usuários e desenvolvedores.
- Embora sejam permitidas alterações que quebrem a ABI do usuário, a ABI não deve ser quebrada gratuitamente. Alterações significativas da ABI do usuário devem ser restritas às versões principais.
- Novos recursos que ainda não foram implementados nas arquiteturas de Tier 2 devem fornecer um meio de desativá-los nessas arquiteturas.

19.5. Tier 3: Arquiteturas Experimentais

As plataformas de Tier 3 têm pelo menos suporte parcial ao FreeBSD. Elas *não* são suportadas pelas equipes de segurança, engenharia de release e de gerenciamento de ports.

As plataformas de Tier 3 são arquiteturas nos estágios iniciais de desenvolvimento, para plataformas de hardware não convencionais, ou que são considerados sistemas legados improváveis de serem amplamente utilizados no futuro. O suporte inicial para plataformas de Tier 3 pode existir em um repositório separado em vez do repositório de código fonte principal.

O Projeto FreeBSD não oferece garantias aos consumidores das plataformas de Tier 3 e não está comprometido em manter recursos para apoiar o seu desenvolvimento. As plataformas de Tier 3 nem sempre podem ser compiláveis, nem quaisquer ABIs do kernel ou de usuário são consideradas

estáveis.

19.6. Tier 4: Arquiteturas Não Suportadas

As plataformas Tier 4 não são suportadas de nenhuma forma pelo projeto.

Todos os sistemas não classificados de outra forma são sistemas de Tier 4. Quando uma plataforma faz a transição para o Tier 4, todo o suporte para a plataforma é removido das árvores de código fonte e de ports. Observe que o suporte aos ports deve ser mantido enquanto a plataforma for suportada em uma branch suportada pelo ports.

19.7. Política para Alteração do Tier de uma Arquitetura

Os sistemas só podem ser movidos de um Tier para outro por meio da aprovação do Core Team do FreeBSD, que deve tomar essa decisão em colaboração com as equipes de segurança, engenharia de release e gerenciamento dos ports. Para que uma plataforma seja promovida para um Tier superior, todas as garantias de suporte ausentes devem ser atendidas antes que a promoção seja concluída.

20. FAQ específico dos Ports

20.1. Como eu adiciono um novo port?

Adicionando um Novo Port

Primeiro, por favor leia a seção sobre cópias do repositório.

A maneira mais fácil de adicionar um novo port é o script `addport` localizado no diretório ports/Tools/scripts. Ele adiciona um port do diretório especificado, determinando a categoria automaticamente a partir do Makefile do port. Também adiciona uma entrada à categoria do Makefile do port. Foi escrito por Michael Haro mharo@FreeBSD.org, Will Andrews will@FreeBSD.org e Renato Botelho garga@FreeBSD.org. Ao enviar perguntas sobre este script para a lista de discussão dos ports do FreeBSD, por favor também copie Chris Rees crees@FreeBSD.org, o mantenedor atual.

20.2. Existe qualquer outra coisa que preciso saber quando adiciono um novo port?

Verifique o port, de preferência para garantir que ele seja compilado e empacotado corretamente. Essa é a seqüência recomendada:

```
# make install
# make package
# make deinstall
```

```
# pkg add package you built above
# make deinstall
# make reinstall
# make package
```

O Manual de Porters contém instruções mais detalhadas.

Use o portlint(1) para verificar a sintaxe do port. Você não precisa necessariamente eliminar todos os avisos, mas certifique-se de ter corrigido os mais simples.

Se o port veio de um remetente que não contribuiu para o projeto antes, adicione o nome dessa pessoa a seção Colaboradores Adicionais da Lista de Colaboradores do FreeBSD.

Feche o PR se o port entrou como um PR. Para fechar um PR, mude o estado para `Issue Resolved` e a resolução como `Fixed`.

21. Problemas Específicos para Desenvolvedores Que Não São Committers

Algumas pessoas que têm acesso às máquinas do FreeBSD não possuem commit bits. Quase todo este documento também será aplicado a esses desenvolvedores (exceto itens específicos de commits e associações de listas de discussão que os acompanham). Em particular, recomendamos que você leia:

- Detalhes Administrativos
- Para todos



Peça ao seu mentor para adicioná-lo em "Colaboradores Adicionais" (doc/en_US.ISO8859-1/articles/contributors/contrib.additional.xml), se você ainda não estiver listado há.

- Relações entre os Desenvolvedores
- Guia de início rápido do SSH
- A Grande Lista de Regras dos Committers do FreeBSD == Informações sobre o Google Analytics

A partir de 12 de dezembro de 2012, o Google Analytics foi habilitado no site do Projeto FreeBSD para coletar estatísticas anônimas sobre o uso do site. As informações coletadas são valiosas para o Projeto de Documentação do FreeBSD, para identificar vários problemas no site do FreeBSD.

21.1. Política Geral do Google Analytics

O projeto FreeBSD leva a privacidade do visitante muito a sério. Como tal, o site do Projeto FreeBSD honra o cabeçalho "Do Not Track"*antes* de buscar o código de monitoramento do Google. Para mais informações, consulte a Política de Privacidade do FreeBSD.

O acesso do Google Analytics *não* é arbitrariamente permitido - o acesso deve ser solicitado, votado

pela Equipe de Engenharia de Documentação doceng@FreeBSD.org e explicitamente concedido.

Solicitações de acesso aos dados do Google Analytics devem incluir uma finalidade específica. Por exemplo, uma razão válida para solicitar acesso seria "para ver os navegadores web usados com mais freqüência pelos visitantes do website do FreeBSD para garantir que as velocidades de renderização da página sejam aceitáveis."

Por outro lado, "ver quais navegadores são usados com mais freqüência" (sem declarar *por que*) seria rejeitado.

Todas as solicitações devem incluir o período de tempo para o qual os dados seriam requisitados. Por exemplo, deve ser explicitamente declarado se os dados solicitados seriam requisitados em um período de tempo que abranja um período de 3 semanas, ou se o pedido seria para acessar apenas uma vez.

Qualquer pedido de dados do Google Analytics sem uma razão clara e razoável que beneficie o Projeto FreeBSD será rejeitado.

21.2. Dados disponíveis por meio do Google Analytics

Alguns exemplos dos tipos de dados do Google Analytics disponíveis incluem:

- Navegadores Web comumente usados
- Tempos de carregamento de páginas
- Acesso ao site por idioma

22. Perguntas Diversas

22.1. Como adiciono um novo arquivo a uma branch?

Para adicionar um arquivo a uma branch, simplesmente faça o check-out ou atualize-o na branch que você deseja adicionar e, em seguida, adicione o arquivo usando a operação add como faria normalmente. Isso funciona bem para as árvores `doc` e `ports`. A árvore `src` usa o SVN e requer mais cuidado por causa das propriedades `mergeinfo`. Veja o Subversion Primer para detalhes sobre como executar um MFC.

22.2. Como eu acesso people.FreeBSD.org para colocar informações pessoais ou de projeto?

O servidor `people.FreeBSD.org` é o mesmo que `freefall.FreeBSD.org`. Basta criar um diretório public_html. Qualquer coisa que você colocar nesse diretório será automaticamente visível em https://people.FreeBSD.org/.

22.3. Onde estão armazenados os arquivos da lista de discussão?

As listas de discussão são arquivadas em /local/mail na `freefall.FreeBSD.org`.

22.4. Eu gostaria de ser mentor de um novo committer. Qual processo eu preciso seguir?

Consulte o documento Novo Procedimento para Criação de Conta nas páginas internas.

23. Benefícios e vantagens para os Commiters do FreeBSD

23.1. Reconhecimento

O reconhecimento como engenheiro de software competente é o valor mais duradouro. Além disso, ter a chance de trabalhar com algumas das melhores pessoas que todos os engenheiros sonham em conhecer é um grande privilégio!

23.2. FreeBSD Mall

Os committers do FreeBSD podem obter um conjunto gratuito de 4-CDs ou DVDs da FreeBSD Mall, Inc. em conferências.

23.3. IRC

Além disso, os desenvolvedores podem solicitar um "cloaked hostmask" para sua conta na rede Freenode de IRC na forma de `freebsd/developer/nome_na_freefall` ou `freebsd/developer/nome_no_NickServ`. Para solicitar uma máscara, envie um e-mail para irc@FreeBSD.org com o nome da sua hostmask solicitada e nome da conta no NickServ.

23.4. `Gandi.net`

A Gandi fornece hospedagem de sites, computação em nuvem, registro de domínio e serviços de certificados X.509.

A Gandi oferece um desconto E-rate para todos os desenvolvedores do FreeBSD. Envie e-mail para non-profit@gandi.net usando o seu endereço de e-mail `@freebsd.org` e indique o seu identificador no Gandi.