Engenharia de Release do FreeBSD

Glen Barber, The FreeBSD Foundation Rubicon Communications, LLC (Netgate) <gjb@FreeBSD.org >

2019-11-21 22:55:26 por dbaio.

Resumo

Este artigo descreve o processo por trás do modelo de engenharia de release adotado pelo Projeto FreeBSD.

Índice

1. Introdução ao Processo de Engenharia de Release do FreeBSD

O desenvolvimento do FreeBSD segue um fluxo muito específico. Em geral, todas as mudanças no sistema base do FreeBSD são feitas em uma branch chamada `head/`, a qual reflete o topo da árvore de código fonte.

Após um período razoável de testes, as alterações podem ser fundidas na branch `stable/`. O período de tempo mínimo padrão antes da fusão das alterações na branch `stable/` é de três (3) dias.

Embora seja uma regra geral esperar pelo menos três (3) dias antes de fundir o código produzido na branch `head/`, existem algumas circunstâncias especiais em que uma fusão imediata pode ser necessária, tal como uma correção de segurança crítica ou uma correção de bug que inibe diretamente o processo de compilação de uma release.

Após vários meses, quando o número de mudanças na branch `stable/` cresceu significativamente, é hora de lançar a próxima versão do FreeBSD. Essas versões foram historicamente chamadas de “point” releases.

Entre as versões das branches `stable/`, aproximadamente a cada dois (2) anos, uma nova versão é criada vinda diretamente da branch `head/`. Essas versões foram historicamente chamadas de versões “dot-zero”.

Este artigo irá destacar o fluxo de trabalho e as responsabilidades da Equipe de Engenharia de Release do FreeBSD para ambas as versões “dot-zero” e “point release”.

As seções a seguir deste artigo descrevem:

Seção 2, “Informação Geral e Preparativos”
: Informações gerais e preparativos antes de iniciar o ciclo de release.

Seção 4, “Alterações na Página Web Durante o Ciclo de Release”
: Alterações na Página Web Durante o Ciclo de Release

Seção 3, “Terminologia da Engenharia de Release”
: Terminologia e informações gerais, como “code slush” e “code freeze”, usadas por todo este documento.

Seção 5, “Versões oriundas da branch `head/`”
: O processo de Engenharia de Release para uma versão “dot-zero”.

Seção 6, “Release a partir da branch `stable/`”
: O processo de Engenharia de Release para uma versão “point release”.

Seção 7, “Construindo a Mídia de Instalação do FreeBSD”
: Informações relacionadas aos procedimentos específicos para construir o meio de instalação.

Seção 8, “Publicando a Mídia de Instalação do FreeBSD nos Espelhos do Projeto”
: Procedimentos para publicar um meio de instalação.

Seção 9, “Encerrando o Ciclo de Release”
: Encerrando o ciclo de release.

2. Informação Geral e Preparativos

Aproximadamente dois meses antes do início do ciclo de vida de uma nova versão, a Equipe de Engenharia de Release do FreeBSD decide sobre um cronograma para o seu lançamento. A programação inclui os vários milestones do ciclo de release, como datas de congelamento, datas para as branches e datas para compilação do código. Por exemplo:

Milestone	Data prevista
pré congelamento da `head/`:	27 de maio de 2016
Congelamento da `head/`:	10 de junho de 2016
Congelamento de KBI da `head/`:	24 de junho de 2016
Pré congelamento da árvore `doc/` [1]:	24 de junho de 2016
Branch trimestral dos ports [2]:	1º de julho de 2016
branch `stable/`*`12/`*:	8 de julho de 2016
Aplicação da tag na árvore `doc/` [3]:	8 de julho de 2016
Começa a compilação da fase BETA1:	8 de julho de 2016
descongelamento da branch `head/`:	9 de julho de 2016
Começa a compilação da fase BETA2:	15 de julho de 2016
Começa a compilação da fase BETA3 [*]:	22 de julho de 2016
branch `releng/`*`12.0/`*:	29 de julho de 2016
A compilação da fase RC1 é iniciada:	29 de julho de 2016
descongelamento da branch `stable/`*`12/`*:	30 de julho de 2016
Começa a compilação da fase RC2:	5 de agosto de 2016
Última compilação dos ports [4]:	6 de agosto de 2016

Milestone	Data prevista
Aplicação da tag na árvore dos ports:	12 de agosto de 2016
compilação da fase RC3 [*]:	12 de agosto de 2016
início de compilação da RELEASE:	19 de agosto de 2016
Anúncio da RELEASE:	2 de setembro de 2016

Nota

Itens marcados com "[*]" identificam passos executados apenas "conforme necessário".

1. O pré congelamento da árvore `doc` é coordenado pela Equipe de Engenharia de Documentação do FreeBSD.
2. A branch trimestral da árvore da coleção de ports é determinada quando a compilação final da fase `RC` é planejada. Uma nova branch trimestral é criada no primeiro dia do trimestre, portanto, essa métrica deve ser usada ao considerar os marcos do ciclo de release. Uma branch trimestral é criada pela Equipe de Gerenciamento de Ports do FreeBSD.
3. A árvore `doc` é recebe a tag da Equipe de Engenharia de Documentação do FreeBSD.
4. A compilação final dos pacotes é feita pela Equipe de Gerenciamento de Ports do FreeBSD após a compilação final (ou o que é esperada ser a final) de uma fase `RC`.

Nota

Se a versão RELEASE estiver sendo criada a partir de uma branch `stable` existente, a data de congelamento do KBI poderá ser excluída, já que o KBI já está congelado em branchs `stable`.

Ao escrever o cronograma do ciclo de versões, várias coisas precisam ser levadas em consideração, em particular os milestones nos quais a data alvo depende de milestones pré-definidos sobre os quais existe uma dependência. Por exemplo, a aplicação da tag de release da Coleção de Ports é originada da branch trimestral ativa no momento da última fase do `RC`. Isso em parte define qual branch trimestral é usada, quando a aplicação da tag pode acontecer e qual revisão da árvore de ports é usada para a construção final de uma `RELEASE`.

Depois de um acordo geral sobre o cronograma, a Equipe de Engenharia de Release do FreeBSD envia e-mails para os desenvolvedores do FreeBSD.

É normal que muitos desenvolvedores informem a Equipe de Engenharia de Release do FreeBSD sobre vários trabalhos em andamento. Em alguns casos, uma extensão para o trabalho em andamento será solicitada e, em outros casos, uma solicitação para uma “aprovação geral” para um subconjunto específico da árvore será feita.

Quando tais solicitações são feitas, é importante certificar-se de que os cronogramas (mesmo que estimados) sejam discutidos. Para as aprovações gerais, o período de tempo para a aprovação geral deve ser claro. Por exemplo, um desenvolvedor do FreeBSD pode solicitar aprovações gerais desde o início do code slush até o início da construção da primeira `RC`.

Nota

Para manter o controle das aprovações gerais, a Equipe de Engenharia de Release do FreeBSD usa um repositório interno para manter um registro de tais solicitações, que define a área na

qual uma aprovação geral foi concedida, o(s) autor(es), quando a aprovação geral expira e a razão pela qual a aprovação foi concedida. Um exemplo disso é a concessão de uma aprovação geral na `release/doc/` a todos os membros da Equipe de Engenharia de Release do FreeBSD até o `RC` final para atualizar as notas de lançamento e outras documentação relacionada ao lançamento.

Nota

A Equipe de Engenharia de Release do FreeBSD também usa este repositório para rastrear solicitações de aprovação pendentes que são recebidas antes de iniciar várias compilações durante o ciclo de release, que o Engenheiro de Release especifica o período de corte com um email para os desenvolvedores do FreeBSD.

Dependendo do conjunto de código subjacente em questão, e do impacto geral que o conjunto de código tem no FreeBSD como um todo, tais solicitações podem ser aprovadas ou negadas pela Equipe de Engenharia de Release do FreeBSD.

O mesmo se aplica às extensões de trabalho em andamento. Por exemplo, o trabalho em andamento para um novo driver de dispositivo que de outra forma é isolado do restante da árvore pode receber uma extensão. Um novo scheduler, no entanto, pode não ser viável, especialmente se tais mudanças dramáticas não existirem em outra branch.

O cronograma também é adicionado ao site do projeto, no repositório `doc`, em `head/en_US.ISO8859-1/htdocs/releases/ 12.0R/schedule.xml` . Este arquivo é continuamente atualizado conforme o ciclo progride.

Nota

Na maioria dos casos, o `schedule.xml` pode ser copiado de uma versão anterior e atualizado de acordo.

Além de adicionar o `schedule.xml` ao site, o `head/share/xml/navibar.ent` e o `head/share/xml/release.ent` também são atualizados para adicionar o link para o cronograma em várias subpáginas, bem como para habilitar o link para o cronograma na página principal do website do projeto.

O cronograma também chamado a partir de `head/en_US.ISO8859-1/htdocs/releng/index.xml` .

Aproximadamente um mês antes do “code slush”, a Equipe de Engenharia de Release do FreeBSD envia um email de lembrete para os desenvolvedores do FreeBSD.

Uma vez que as primeiras compilações do ciclo de release estejam disponíveis, atualize a entidade `beta.local.where` em `head/en_US.ISO8859-1/htdocs/releases/ 12.0R/schedule.xml` . substituindo `IGNORE` por `INCLUDE` .

Nota

Se dois ciclos de lançamento paralelo estão acontecendo ao mesmo tempo, a entidade `beta2.local.where` pode ser usada no lugar.

3. Terminologia da Engenharia de Release

Esta seção descreve algumas das terminologias usadas no restante deste documento.

3.1. O Code Slush

Embora o code slush não seja um congelamento mandatório da árvore, a Equipe de Engenharia de Release do FreeBSD solicita que resoluções dos bugs existentes no código tenham prioridade sobre implementação de novos recursos.

O code slush não impõe aprovações de confirmação para o Branch.

3.2. O Code Freeze

O code freeze marca o momento em que todos os commits para a branch exigem aprovação explícita da Equipe de Engenharia de Release do FreeBSD.

O repositório Subversion do FreeBSD contém vários ganchos para executar verificações de integridade antes que qualquer commit seja realmente confirmado na árvore. Um desses ganchos avaliará se o comprometimento com uma branch específica requer aprovação específica.

Para impor aprovações de commit pela Equipe de Engenharia de Release do FreeBSD, o Engenheiro de Release atualiza o `base/svnadmin/conf/approvers` , e aplica a mudança de volta para o repositório. Feito isso, qualquer alteração na branch deve incluir uma linha “Aprovado por:” na mensagem de commit.

A linha “Aprovada por:” deve corresponder à segunda coluna em `base/svnadmin/conf/aprovovers` , caso contrário, o commit será rejeitado pelos hooks do repositório.

Nota

Durante o code freeze, os committers do FreeBSD devem seguir as Recomendações de Requisição de Mudanças.

3.3. O KBI / Congelamento KPI

A estabilidade de KBI/KPI implica que o caller (que faz uma chamada) de uma função através de duas versões diferentes de software que implementam a função, resulta no mesmo estado final. O caller, seja um processo, thread ou função, espera que a função opere de uma determinada maneira, caso contrário, a estabilidade do KBI/KPI na branch é interrompida.

4. Alterações na Página Web Durante o Ciclo de Release

Esta seção descreve as alterações no site que devem ocorrer conforme o ciclo de lançamento progride.

Nota

Os arquivos especificados ao longo desta seção são relativos à branch `head/` do repositório `doc` no Subversion.

4.1. Alterações na Página Web Antes do Início do Ciclo de Release

Quando o cronograma do ciclo de release está disponível, esses arquivos precisam ser atualizados para habilitar várias funcionalidades diferentes no site do Projeto FreeBSD:

Arquivo para editar	O que mudar
`share/xml/release.ent`	Altere `beta.upcoming` de `IGNORE` para `INCLUDE`
`share/xml/release.ent`	Altere `% beta.upcoming` de `IGNORE` para `INCLUDE`
`share/xml/release.ent`	Altere `beta.testing` de `IGNORE` para `INCLUDE`
`share/xml/release.ent`	Altere `% beta.testing` de `IGNORE` para `INCLUDE`

4.2. Alterações na página web durante a fase BETA ou RC

Ao fazer a transição de `PRERELEASE` para `BETA`, esses arquivos precisam ser atualizados para ativar o bloco "Teste de ajuda" na página de download. Todos os arquivos são relativos ao `head/` no repositório `doc`:

Arquivo para editar	O que mudar
`en_US.ISO8859-1/htdocs/releases/12.0R/schedule.xml`	Altere `% beta.local.where IGNORE` para `INCLUDE`
`share/xml/release.ent`	Atualize `% betarel.vers` para `BETA`*`1`*
`share/xml/news.xml`	Adicione uma entrada anunciando a versão `BETA`
`en_US.ISO8859-1/htdocs/security/advisory-template.txt`	Adicione as novas `BETA`, `RC` ou `RELEASE` final ao modelo
`en_US.ISO8859-1/htdocs/security/errata-template.txt`	Adicione as novas `BETA`, `RC` ou `RELEASE` final ao modelo

Uma vez criada a branch `releng/`*`12.0`*`/`, os diversos documentos relacionados à release precisam ser gerados e adicionados manualmente ao repositório `doc/`.

Dentro de `release/doc` , invoque make(1) para gerar as páginas `errata.html` , `hardware.html` , `readme.html` e `relnotes.html` , que são então adicionadas ao diretório `doc/head/en_US.ISO8859-1/htdocs/releases/` *`XY`*`R/`, em que *`XY`* representa o número da versão principal e da versão secundária.

A propriedade `fbsd:nokeywords` deve ser definido como `on` nos arquivos recém-adicionados para que os hooks de pré-commit permitam que eles sejam adicionados ao repositório.

Nota

Os documentos relevantes relacionados à release existem no repositório `doc` para FreeBSD 12.x e posterior.

4.3. Mudanças nos ports durante as fases BETA, RC, e a versão RELEASE final

Para cada compilação durante o ciclo de release, os arquivos `MANIFEST` contendo o `SHA256` dos vários conjuntos de distribuição, como `base.txz` , `kernel.txz` , e assim por diante, são adicionados ao port misc/freebsd-release-manifests. Isso permite outros utilitários além do bsdinstall(8), como ports-mgmt/poudriere, usem esses conjuntos de distribuição com segurança fornecendo um mecanismo através do qual os checksums possam ser verificados.

5. Versões oriundas da branch head/

Esta seção descreve os procedimentos gerais do ciclo de release do FreeBSD na branch `head`.

5.1. Compilações "ALPHA" do FreeBSD

Tendo aparecido primeiramente durante o ciclo de release do FreeBSD 10.0-RELEASE, a noção de compilações de fases "ALPHA" foi introduzida. Ao contrário das compilações BETA e RC, as compilações desse novo estágio ALPHA não fazem parte do cronograma de Release do FreeBSD.

A idéia por trás das compilações ALPHA é disponibilizar builds regulares fornecidas pelo FreeBSD antes da criação da branch stable/.

Os snapshots ALPHA do FreeBSD devem ser preparados aproximadamente uma vez por semana.

Para a primeira compilação ALPHA, o valor BRANCH em sys/conf/newvers.sh precisa ser alterado de CURRENT para ALPHA1. Para compilações ALPHA subsequentes, incremente cada valor de ALPHA*N* em um.

Veja Seção 7, "Construindo a Mídia de Instalação do FreeBSD" para informações sobre como construir as imagens ALPHA.

5.2. Criando a branch stable/*12*/

Ao criar a branch stable/, várias alterações são necessárias na nova branch stable/ e na branch head/. Os arquivos listados são relativos ao repositório raiz. Para criar a nova branch stable/*12*/ no Subversion:

```
% svn cp ^/head stable/12/
```

Uma vez que a branch stable/*12*/ tenha sido criada, faça as seguintes edições:

Arquivo para editar	O que mudar
stable/*12*/UPDATING	Atualize a versão do FreeBSD e remova o aviso sobre WITNESS

stable/*12*/contrib/jemalloc/include/jemalloc/jemalloc_FreeBSD.h	`#ifndef MALLOC_PRODUCTION`
	`#define MALLOC_PRODUCTION`
	`#endif`

stable/*12*/lib/clang/llvm.build.mk	Remova o comentário -DNDEBUG
stable/*12*/sys/*/conf/GENERIC*	Remova o suporte de depuração
stable/*12*/sys/*/conf/MINIMAL	Remova o suporte de depuração
stable/*12*/release/release.conf.sample	Atualize o SRCBRANCH
stable/*12*/sys/*/conf/GENERIC-NODEBUG	Remova essas configurações do kernel
stable/*12*/sys/arm/conf/std.arm*	Remova as opções de depuração
stable/*12*/sys/conf/newvers.sh	Atualize o valor de BRANCH para refletir BETA1
stable/*12*/share/mk/src.opts.mk	Mova REPRODUCIBLE_BUILD de __DEFAULT_NO_OPTIONS para __DEFAULT_YES_OPTIONS
stable/*12*/libexec/rc/rc.conf	Defina o dumpdev de AUTO para NO (ele é configurável via bsdinstall(8) para aqueles que o querem habilitado por padrão)
stable/*12*/release/Makefile	Remova as entradas debug.witness.trace

Então, na branch head/, que agora se tornará uma nova versão principal:

Arquivo para editar	O que mudar
head/UPDATING	Atualize a versão do FreeBSD

Arquivo para editar	O que mudar
head/sys/conf/newvers.sh	Atualize o valor de BRANCH para refletir CURRENT e incremente a REVISION
head/Makefile.inc1	Atualize o TARGET_TRIPLE e o MACHINE_TRIPLE
head/sys/sys/param.h	Atualize o __FreeBSD_version
head/gnu/usr.bin/cc/cc_tools/freebsd-native.h	Atualize o FBSD_MAJOR e o FBSD_CC_VER
head/contrib/gcc/config.gcc	Anexe a seção freebsd<versão>.h
head/lib/clang/llvm.build.mk	Atualize o valor do OS_VERSION
head/lib/clang/freebsd_cc_version.h	Atualize o FREEBSD_CC_VERSION
head/lib/clang/include/lld/Common/Version.inc	Atualize o LLD_REVISION_STRING
head/Makefile.libcompat	Atualize o LILB32CPUFLAGS

6. Release a partir da branch stable/

Esta seção descreve os procedimentos gerais do ciclo de release do FreeBSD a partir de uma branch stable/.

6.1. Code Slush da branch stable do FreeBSD

Na preparação para o code freeze em uma branch stable, vários arquivos precisam ser atualizados para refletir o ciclo de release que está oficialmente em andamento. Esses arquivos são todos relativos ao nível mais alto da branch stable:

Arquivo para editar	O que mudar
sys/conf/newvers.sh	Atualize o valor da BRANCH para refletir PRERELEASE
Makefile.inc1	Atualize o TARGET_TRIPLE
lib/clang/llvm.build.mk	Atualize o OS_VERSION
Makefile.libcompat	Atualize o LILB32CPUFLAGS
gnu/usr.bin/groff/tmac/mdoc.local.in	Adiciona uma nova entrada .ds para a versão do FreeBSD, e atualiza doc-default-operating-system (FreeBSD 11.x e anteriores apenas)

No repositório doc, atualize também head/pt_BR.ISO8859-1/htdocs/releases/*12.0*R/Makefile.hardware, alternando o valor de _BRANCH para BETA*X*, RC*X* ou RELEASE, respectivamente.

6.2. Builds BETA do FreeBSD

Após o code slush, a próxima fase do ciclo de release é o code freeze. Este é o ponto no qual todos os commits para a branch stable requerem aprovação explícita da Equipe de Engenharia de Release do FreeBSD. Isto é reforçado por hooks de pré-commit no repositório Subversion editando base/svnadmin/conf/approvers para incluir uma expressão regular que coincida com a branch stable/*12*/ para a release:

```
^/stable/12/ re
^/releng/12.0/ re
```

Nota

Há duas exceções gerais para exigir aprovação de commit durante o ciclo de release. A primeira é qualquer alteração que precise ser "committed" pelo Engenheiro de Release para

continuar com o fluxo de trabalho diário do ciclo de lançamento, e a outra são as correções de segurança que podem ocorrer durante o ciclo de lançamento.

Quando o code freeze estiver em vigor, a próxima construção da branch será rotulada como `BETA1`. Isso é feito atualizando o valor de `BRANCH` em `sys/conf/newvers.sh` de `PRERELEASE` para `BETA1`.

Feito isso, o primeiro conjunto de builds `BETA` é iniciado. Builds `BETA` subseqüentes não requerem atualizações em nenhum arquivo diferente do `sys/conf/newvers.sh`, incrementando o número de compilação da versão `BETA`.

6.3. Criando a branch `releng/12.0/`

Quando a primeira construção `RC` (Release Candidate) está pronta para começar, a branch `releng/` é criada. Este é um processo de várias etapas que deve ser feito em uma ordem específica, a fim de evitar anomalias, como sobreposições com valores de `__FreeBSD_version`, por exemplo. Os caminhos listados abaixo são relativos ao repositório raiz. A ordem dos commits e o que mudar são:

```
% svn cp ^/stable/12/ releng/12.0/
```

Arquivo para editar	O que mudar
`releng/12.0/sys/conf/newvers.sh`	Altere `BETA`*X* para `RC1`
`releng/12.0/sys/sys/param.h`	Atualize o `__FreeBSD_version`
`releng/12.0/etc/pkg/FreeBSD.conf`	Substitua `latest` por `quarterly` (trimestral) como a localização padrão do repositório de pacotes
`releng/12.0/release/pkg_repos/release-dvd.conf`	Substitua `latest` por `quarterly` (trimestral) como a localização padrão do repositório de pacotes
`stable/12/sys/conf/newvers.sh`	Atualize `BETA`*X* para `PRERELEASE`
`stable/12/sys/sys/param.h`	Atualize o `__FreeBSD_version`
`svnadmin/conf/approvers`	Adicione uma nova linha de aprovadores para a branch releng como foi feito para a branch stable

```
% svn propdel -R svn:mergeinfo  releng/12.0/
% svn commit  releng/12.0/
% svn commit  stable/12/
```

Agora que existem dois novos valores de `__FreeBSD_version`, também atualize `head/pt_BR.ISO8859-1/books/porters-handbook/versions/chapter.xml` no repositório do Projeto de Documentação.

Depois que a primeira compilação de um `RC` estiver concluída e testada, a branch `stable/` pode ser "descongelada" removendo (ou comentando) a entrada `^/stable/12/` em `svnadmin/conf/approvers`.

Seguindo a disponibilidade do primeiro `RC`, o Time Bugmeister do FreeBSD deve ser avisado por e-mail para adicionar o novo FreeBSD `-RELEASE` às `versões` disponíveis no menu drop-down exibido no rastreador de bugs.

7. Construindo a Mídia de Instalação do FreeBSD

Esta seção descreve os procedimentos gerais de produção de snapshots e releases de desenvolvimento do FreeBSD.

7.1. Scripts para compilação de Releases

Esta seção descreve os scripts de build usados pela Equipe de Engenharia de Release do FreeBSD para produzir snapshots da versão em desenvolvimento e das releases.

7.1.1. O script `release.sh`

Antes do FreeBSD 9.0-RELEASE, o `src/release/Makefile` era atualizado para suportar o bsdinstall(8), e o script `src/release/generate-release.sh` foi introduzido como um wrapper para automatizar a chamada dos targets release(7).

Antes do FreeBSD 9.2-RELEASE, foi introduzido o `src/release/release.sh`, que baseado fortemente em `src/release/generate-release.sh` incluía suporte para especificar arquivos de configuração para substituir várias opções e variáveis de ambiente. O suporte para arquivos de configuração forneceu suporte para cross building (compilação para mais de uma arquitetura) de uma release para cada arquitetura, especificando um arquivo de configuração separado para cada chamada.

Como um breve exemplo do uso de `src/release/release.sh` para construir uma única versão em `/scratch`:

```
# /bin/sh /usr/src/release/release.sh
```

Como um breve exemplo do uso de `src/release/release.sh` para construir uma única versão cross-build (entre arquiteturas) usando um diretório de destino diferente, crie um `release.conf` personalizado contendo:

```
# release.sh configuration for powerpc/powerpc64
CHROOTDIR="/scratch-powerpc64"
TARGET="powerpc"
TARGET_ARCH="powerpc64"
KERNEL="GENERIC64"
```

Em seguida, invoque `src/release/release.sh` da seguinte forma:

```
# /bin/sh /usr/src/release/release.sh -c /HOME/release.conf
```

Veja release(7) e `src/release/release.conf.sample` para mais detalhes e exemplos de uso.

7.1.2. O Script Wrapper `thermite.sh`

Para tornar o cross building do conjunto completo de arquiteturas suportadas em uma determinada branch mais rápido, mais fácil e reduzindo os fatores de erro humano, um script wrapper de apoio ao `src/release/release.sh` foi escrito para iterar pelas várias combinações de arquiteturas e chamar o script `src/release/release.sh` usando um arquivo de configuração específico para essa arquitetura.

O script wrapper é chamado de `thermite.sh`, o qual está disponível no repositório Subversion do FreeBSD em `svn://svn.freebsd.org/base/user/gjb/thermite/`, além dos arquivos de configuração usados para construir os snapshots de desenvolvimento `head/` e `stable/`*`12/`*.

O uso do `thermite.sh` é explicado em Seção 7.2, “Construindo Snapshots de Desenvolvimento do FreeBSD” e Seção 7.3, “Construindo Releases do FreeBSD”.

Cada arquitetura e kernel individual tem seu próprio arquivo de configuração usado pelo `release.sh`. Cada branch tem sua própria configuração `defaults-X.conf` que contém entradas comuns em cada arquitetura, onde substituições ou variáveis especiais são definidas e/ou substituídas nos arquivos por compilação.

O esquema de nomenclatura do arquivo de configuração por compilação está na forma de `${revision}-${TARGET_ARCH}-${KERNCONF}-${type}.conf`, em que as variáveis em maiúsculas são equivalentes a que make(1) usa no sistema de compilação e as variáveis minúsculas são definidas nos arquivos de configuração, mapeando para a versão principal da respectiva branch.

Cada branch também possui sua própria configuração `builds-X.conf`, que é usada pelo `thermite.sh`. O script `thermite.sh` itera através de cada valor ${revision}, ${TARGET_ARCH}, ${KERNCONF} e ${type}, criando uma lista principal do que construir. No entanto, uma determinada combinação da lista só será criada se o respectivo arquivo de configuração existir, que é onde o esquema de nomenclatura acima é relevante.

Existem dois caminhos de fornecimento de arquivos:

- `builds-`*`12`*`.conf` -> `main.conf`

  Isto controla o comportamento do `thermite.sh`

- *`12-amd64-GENERIC-snap`*`.conf` -> `defaults-`*`12`*`.conf` -> `main.conf`

  Isto controla o comportamento do `release/release.sh` dentro do chroot(8) de compilação

Nota

Os arquivos de configuração `builds-`*`12`*`.conf`, `defaults-`*`12`*`.conf`, e `main.conf` existem para reduzir a repetição entre os vários arquivos por compilação.

7.2. Construindo Snapshots de Desenvolvimento do FreeBSD

As máquinas oficiais de compilação de versões têm um layout do sistema de arquivos específico, que utiliza ZFS, o `thermite.sh` tira grande proveito de clones e snapshots, garantindo um ambiente de compilação uniforme e consistente.

Os scripts de compilação localizam-se respectivamente em `/releng/scripts-snapshot/scripts` ou `/releng/scripts-release/scripts` , para evitar colisões entre uma compilação `RC` de uma branch releng contra um snapshot `STABLE` da respectiva branch stable.

Existe um dataset (conjunto de dados) separado para as imagens finais de compilação, `/snap/ftp` . Este diretório contém diretórios de snapshots e releases. Eles são usados apenas se a variável `EVERYTHINGISFINE` estiver definida em `main.conf` .

Nota

O nome da variável `EVERYTHINGISFINE` foi escolhido para evitar a colisão com uma variável possivelmente definida no ambiente do usuário, ativando acidentalmente o comportamento que depende de sua definição.

Como o `thermite.sh` percorre a lista principal de combinações e localiza o arquivo de configuração por compilação, um dataset ZFS é criado sob o `/releng` , tal como `/releng/12-amd64-GENERIC-snap` . O checkout das árvores `src/`, `ports/` e `doc/` é realizado em diferentes datasets ZFS, tal como `/releng/12-src-snap` , os quais são então clonados e montados nos respectivos datasets de compilação. Isso é feito para evitar a remoção de uma determinada árvore mais de uma vez.

Assumindo esses caminhos do sistema de arquivos, o `thermite.sh` deveria ser chamado como:

```
# cd /releng/scripts-snapshot/scripts
# ./setrev.sh -b stable/12/
# ./zfs-setup.sh -c ./builds- 12.conf
# ./thermite.sh -c ./builds- 12.conf
```

Quando as compilações forem concluídas, scripts adicionais auxiliares estarão disponíveis para gerar e-mails de snapshots de desenvolvimento que são enviados para a lista de e-mail `freebsd-snapshots@freebsd.org` :

```
# cd /releng/scripts-snapshot/scripts
# ./get-checksums.sh -c ./builds- 12.conf | ./generate-email.pl > snapshot- 12-mail
```

Nota

A saída gerada deve ser checada duas vezes para garantir a exatidão, e o próprio e-mail deve ter assinatura PGP, in-line (no arquivo).

Nota

Esses scripts auxiliares aplicam-se apenas às compilações de snapshot (versão instantânea) de desenvolvimento. Os anúncios durante o ciclo de lançamento (excluindo o anúncio de versão final) são criados a partir de um modelo de email. Uma amostra do modelo de email usado atualmente pode ser encontrada aqui.

7.3. Construindo Releases do FreeBSD

Similar a compilação de snapshots de desenvolvimento do FreeBSD, o `thermite.sh` seria invocado da mesma maneira. A diferença entre snapshots de desenvolvimento e builds de releases, `BETA` e `RC` inclusos, é que os arquivos de configuração do chroot(8) devem ser nomeados com `release` ao invés de `snap` no "type", como mencionado acima.

Além disso, `BUILDTYPE` e `types` devem ser alterados de `snap` para `release` em `defaults-`*`12`*`.conf` e `builds-`*`12`*`.conf`, respectivamente.

Ao construir o `BETA`, o `RC`, e o `RELEASE` final, também ajuste estaticamente o `BUILDSVNREV` para a revisão na branch refletindo a mudança de nome, `BUILDDATE` para a data em que as compilações são iniciadas no formato `YYYYMMDD`. Se as árvores `doc/` e `ports/` tiverem sido marcadas, defina também o `PORTBRANCH` e o `DOCBRANCH` para o caminho da tag relevante no repositório Subversion, substituindo `HEAD` pela última revisão alterada. Também defina `releasesrc` em `builds-`*`12`*`.conf` para a branch relevante, como `stable/`*`12/`* ou `releng/`*`12.0/`*.

Durante o ciclo de release, uma cópia do `CHECKSUM.SHA512` e do `CHECKSUM.SHA256` para cada arquitetura é armazenada no repositório interno da Equipe de Engenharia de Release do FreeBSD, além de ser incluída nos diversos e-mails de anúncio. Cada `MANIFEST` contendo os hashes do `base.txz`, do `kernel.txz`, etc. também são adicionados ao misc/freebsd-release-manifests na coleção de ports.

Depois de construir a `RELEASE` final, a branch `releng/`*`12.0/`* é marcada como `release/`*`12.0.0/`* usando a revisão a partir da qual a `RELEASE` foi construída. Semelhante a criar as branches `stable/`*`12/`* e `releng/`*`12.0/`*, isso é feito com `svn cp`. Da raiz do repositório:

```
% svn cp ^/releng/12.0/@r306420 release/12.0.0/
% svn commit release/12.0.0/
```

8. Publicando a Mídia de Instalação do FreeBSD nos Espelhos do Projeto

Esta seção descreve o procedimento para publicar snapshots e releases de desenvolvimento do FreeBSD nos espelhos do Projeto.

8.1. Preparando Imagens de Mídias de Instalação do FreeBSD

A preparação dos snapshots e das versões do FreeBSD é um processo de duas partes:

- Criando a estrutura de diretórios para corresponder a hierarquia em `ftp-master`

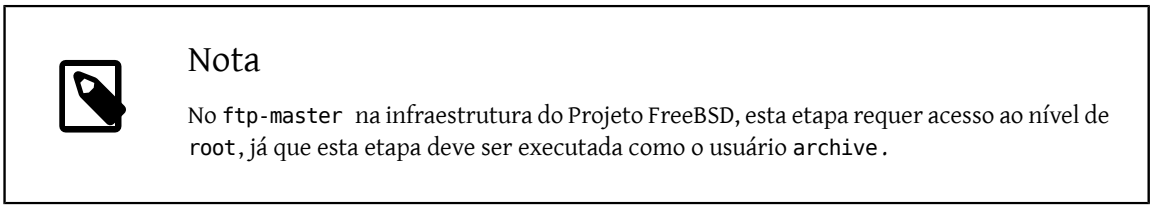
Se `EVERYTHINGISFINE` for definido nos arquivos de configuração de compilação, `main.conf` no caso dos scripts de compilação mencionados acima, isto acontece automaticamente no chroot(8) após a compilação ser concluída, criando a estrutura de diretório em `${DESTDIR}/R/ftp-stage` com um estrutura de caminho que corresponde ao que é esperado em `ftp-master`. Isto é equivalente a executar o seguinte diretamente no chroot(8):

```
# make -C /usr/src/release -f Makefile.mirrors EVERYTHINGISFINE=1 ftp-stage
```

Depois que cada arquitetura é compilada, o `thermite.sh` irá fazer um rsync do `${DESTDIR}/R/ftp-stage` da compilação chroot(8) para o diretório `/snap/ftp/snapshots` ou `/snap/ftp/releases` no host de compilação, respectivamente.

- Copiando os arquivos para um diretório temporário em `ftp-master` antes de mover os arquivos para `pub/` para iniciar a propagação para os servidores espelhos do Projeto

  Uma vez que todas as compilações terminarem, `/snap/ftp/snapshots`, ou `/snap/ftp/releases` para uma versão, é pesquisado pelo `ftp-master` usando rsync para `/archive/tmp/snapshots` ou `/archive/tmp/releases`, respectivamente.



Nota

No `ftp-master` na infraestrutura do Projeto FreeBSD, esta etapa requer acesso ao nível de `root`, já que esta etapa deve ser executada como o usuário `archive`.

8.2. Publicando a Mídia de Instalação do FreeBSD

Uma vez que as imagens são colocadas em `/archive/tmp/`, elas estão prontas para serem publicadas colocando-as em `/archive/pub/FreeBSD`. Para reduzir o tempo de propagação, o pax(1) é usado para criar links físicos a partir de `/archive/tmp` para `/archive/pub/FreeBSD`.

Nota

Para que isto seja efetivo, tanto o `/archive/tmp` quanto o `/archive/pub` devem residir no mesmo sistema de arquivos lógico.

Há uma ressalva, no entanto, em que o rsync deve ser usado após o pax(1) para corrigir os links simbólicos no `pub/FreeBSD/` *`snapshots`* `/ISO-IMAGES` que o pax(1) irá substituir por um hard link, aumentando o tempo de propagação.

Nota

Assim como nas etapas de preparação, isto requer acesso em nível de `root`, já que essa etapa deve ser executada como o usuário `archive`.

Como o usuário `archive`:

```
% cd /archive/tmp/ snapshots
% pax -r -w -l . /archive/pub/FreeBSD/ snapshots
% /usr/local/bin/rsync -avH /archive/tmp/ snapshots /* /archive/pub/FreeBSD/ snapshots /
```

Substitua os *snapshots* por *releases* conforme apropriado.

9. Encerrando o Ciclo de Release

Esta seção descreve as tarefas gerais de pós-release.

9.1. Avisos de Erratas de Pós-Release

A medida que o ciclo de release se aproxima da conclusão, é comum ter vários candidatos a EN (Aviso de Erratas) para abordar os problemas que foram descobertos ao final do ciclo. Após o lançamento, a Equipe de Engenharia de Release do FreeBSD e a Equipe de Segurança do FreeBSD reveem mudanças que não foram aprovadas antes da versão final, e dependendo do escopo da mudança em questão, podem emitir um EN.

Nota

O processo atual de emissão de ENs é tratado pela Equipe de Segurança do FreeBSD.

Para solicitar uma Errata após a conclusão de um ciclo de lançamento, o desenvolvedor deve preencher o Template de Errata>, em particular as seções `Background` , `Problem Description` , `Impact` e, se aplicável, as seções `Workaround` .

O modelo de Errata preenchido deve ser enviado por e-mail juntamente com um patch na branch `releng/` ou uma lista de revisões da branch `stable/` .

Para pedidos de Errata imediatamente após o lançamento, o pedido deve ser enviado por e-mail à Equipe de Engenharia de Releases do FreeBSD e à Equipe de Segurança do FreeBSD. Depois que a branch `releng/` foi entregue à equipe de Segurança do FreeBSD, conforme descrito em Seção 9.2, “Entrega para a Equipe de Segurança do FreeBSD”, as solicitações de Errata devem ser enviadas à equipe de Segurança do FreeBSD.

9.2. Entrega para a Equipe de Segurança do FreeBSD

Aproximadamente duas semanas após o lançamento, o Engenheiro de Release atualiza o `svnadmin/conf/approvers` alterando a coluna do aprovador de `re` para `(so|security-officer)` para a branch `releng/`*12.0*`/`.