Ingeniería de versiones de FreeBSD

Resumen

Este artículo describe el proceso detrás del modelo de ingeniería de versiones adoptado por el Proyecto FreeBSD.

Este documento todavía no ha sido actualizado para describir los procedimientos actuales del equipo de Ingeniería de Versiones de FreeBSD que siguieron a la transición de Subversion a Git.

Tabla de contenidos

1. Introducción al Proceso de Ingeniería de Versiones de FreeBSD . . . 1
2. Información General y Preparación . . . 2
3. Terminología de la Ingeniería de Versiones . . . 5
4. Cambios en el Sitio Web Durante el Ciclo de Liberación . . . 6
5. Versiones desde head/ . . . 7
6. Versiones desde stable/ . . . 9
7. Construir los Medios de Instalación de FreeBSD . . . 11
8. Publicar los Medios de Instalación de FreeBSD para los Mirrors del Proyecto . . . 14
9. Cerrando el Ciclo de Liberación . . . 16
10. Fin del ciclo de Vida de la Versión . . . 16

1. Introducción al Proceso de Ingeniería de Versiones de FreeBSD

El desarrollo de FreeBSD tiene un flujo de trabajo muy específico. En general, todos los cambios en el sistema base de FreeBSD se hacen en la rama head/, que representa la cima del árbol de fuentes.

Después de un periodo de pruebas razonable, los cambios se pueden integrar en las ramas stable/. El tiempo mínimo por defecto para integrar cambios en las ramas stable/ es de tres (3) días.

A pesar de la regla general de esperar un mínimo de tres días antes de integrar cambios desde head/, hay algunas circunstancias especiales bajo las cuales una integración inmediata puede ser necesaria, como arreglos críticos de seguridad, o un arreglo de un fallo que directamente imposibilita el proceso de generación de versiones.

Después de varios meses, y de que el número de cambios en la rama stable/ haya crecido significativamente, es hora de liberar la siguiente versión de FreeBSD. Estas versiones han sido denominadas históricamente como versiones punto ("point" releases).

Entre las versiones de las ramas stable/, aproximadamente cada dos (2) años, se sacará una versión directamente desde head/. Estas versiones se han denominado históricamente como versiones "punto cero" ("dot-zero" releases).

Este artículo resaltará el flujo de trabajo y responsabilidades del FreeBSD Release Engineering Team tanto para las versiones "dot-zero" como las versiones "point".

Las siguientes secciones de este artículo describen:

Información General y Preparación

Información general y preparación antes de comenzar el ciclo de lanzamiento.

Cambios en el Sitio Web Durante el Ciclo de Liberación

Cambios en el Sitio Web Durante el Ciclo de Liberación

Terminología de la Ingeniería de Versiones

Terminología e información general, tales como "code slush" y "code freeze", usados en este documento.

Versiones desde head/

El proceso de Ingeniería de Versiones para una versión "dot-zero".

Versiones desde stable/

El proceso de Ingeniería de Versiones para una versión "point".

Construir los Medios de Instalación de FreeBSD

Información relacionada con los procedimientos específicos para construir el medio de instalación.

Publicar los Medios de Instalación de FreeBSD para los Mirrors del Proyecto

Procedimientos para publicar un medio de instalación.

Cerrando el Ciclo de Liberación

Terminando el ciclo de lanzamiento.

2. Información General y Preparación

Aproximadamente dos meses antes del ciclo de lanzamiento, el FreeBSD Release Engineering Team decide una programación para la versión. La programación incluye varios hitos en el ciclo de lanzamiento como fechas de congelación, fechas para ramificación y fechas para construcción. Por ejemplo:

Hito	Fecha Esperada
head/ slush:	May 27, 2016
head/ freeze:	June 10, 2016
head/ KBI freeze:	June 24, 2016

Hito	Fecha Esperada
`doc/` tree slush [1]:	June 24, 2016
Rama trimestral de ports [2]:	July 1, 2016
rama stable/12/:	July 8, 2016
`doc/` tree tag [3]:	July 8, 2016
Comienzo de la construcción de BETA1:	July 8, 2016
head/ thaw:	July 9, 2016
Comienzo de la construcción de BETA2:	July 15, 2016
Comienzo de la construcción de BETA3 [*]:	July 22, 2016
rama releng/12.0/:	July 29, 2016
Comienzo de la construcción de RC1:	July 29, 2016
rama stable/12/ thaw:	July 30, 2016
Comienzo de la construcción de RC2:	August 5, 2016
Construcción final de paquetes de Ports [4]:	August 6, 2016
Etiqueta de versión de Ports:	August 12, 2016
Comienzo de la construcción de RC3 [*]:	August 12, 2016
Comienzo de la construcción de RELEASE:	August 19, 2016
Anuncio de RELEASE:	September 2, 2016

Los elementos marcados con "[*]" identifican los pasos realizados solo "cuando sean necesarios".

1. El "slush" (semi congelación) del árbol `doc/` está coordinado por el FreeBSD Documentation Engineering Team.
2. La rama trimestral de Ports que va a ser utilizada se determina por el momento en el que se planifica la construcción de la `RC` final. Una nueva rama trimestral es creada el primer día del trimestre, por lo que se debería usar esta métrica cuando se consideren los hitos del ciclo de lanzamiento. La rama trimestral es creada por el FreeBSD Ports Management Team.
3. El árbol `doc/` es etiquetado por el FreeBSD Documentation Engineering Team.
4. La construcción final de los paquetes de Ports es realizada por el FreeBSD Ports Management Team después de la construcción `RC` definitiva (o lo que se espera que sea la definitiva).

Si la versión se está creando a partir de una rama stable/ existente, la fecha de congelación del KBI (Kernel Binary Interface) se puede excluir, ya que el KBI ya se considera congelado en ramas stable/ establecidas.

Al escribir el programa del ciclo de lanzamientos, se deben tener en cuenta varias cosas, en particular, los hitos en los que la fecha objetivo depende de los hitos predefinidos de los que exista una dependencia. Por ejemplo, la etiqueta de la Colección de Ports se obtiene de la rama trimestral que esté activa en el momento de la última fase `RC`. Esto, en parte, define qué rama trimestral se

usa, cuándo se puede realizar la etiqueta y qué revisión del árbol de ports se usa para la compilación final de RELEASE.

Después de un acuerdo general sobre la programación, el FreeBSD Release Engineering Team envía un correo electrónico con la programación a los Desarrolladores de FreeBSD.

Es algo típico que muchos desarrolladores informen al FreeBSD Release Engineering Team sobre varios trabajos en curso. En algunos casos, se solicitará una extensión del trabajo en curso y en otros casos, se hará una petición de "blanket approval" (aprobación total) para un subconjunto particular del árbol.

Cuando se hacen tales solicitudes, es importante asegurarse de que se discutan los plazos (incluso si se estiman). Para las aprobaciones generales, el período de tiempo para la aprobación general debe quedar claro. Por ejemplo, un desarrollador de FreeBSD puede solicitar una aprobación general desde el principio del slush del código hasta el inicio de la compilación de la RC.

> Para realizar el seguimiento de los "blanket approvals", el FreeBSD Release Engineering Team usa un repositorio interno para mantener un registro de esas peticiones, el cual define el área sobre la que se aprueba el "blanket approval", los autores, cuando expira la aprobación y la razón por la que fue concedido. Un ejemplo de esto es aprobar "blanket approval" para release/doc/ para todos los miembros de FreeBSD Release Engineering Team hasta la RC definitiva con el fin de actualizar las notas de la versión y otra documentación relacionada con ella.

> El equipo de ingeniería de versiones de FreeBSD también utiliza este repositorio para rastrear las solicitudes de aprobación pendientes que se reciben justo antes de empezar el ciclo de compilaciones del lanzamiento, que el Ingeniero de Lanzamientos especifica el período límite con un correo electrónico a los desarrolladores de FreeBSD.

Dependiendo del conjunto de código que haya por debajo y el impacto tal que el conjunto de código tienen en FreeBSD como un todo, esas peticiones pueden ser aprobadas o denegadas por el FreeBSD Release Engineering Team.

Lo mismo se aplica al trabajo que haya en progreso. Por ejemplo, en el caso del desarrollo de un nuevo controlador, el cual está aislado del resto del árbol del código fuente, podría tener una extensión de tiempo. Sin embargo, en el caso de un nuevo planificador de procesos, puede no ser viable, en especial si los cambios no están en otra rama.

El programa también se añade a la página web del Proyecto, en el repositorio doc/, en ~/website/content/en/releases/12.0R/schedule.adoc. Este fichero se actualiza continuamente según progresa el ciclo de lanzamiento.

> La mayor parte de las veces, el fichero schedule.adoc se puede copiar de una versión anterior y actualizarlo como corresponda.

Además de añadir schedule.adoc al sitio web, también se actualiza ~/shared/releases.adoc para añadir el enlace al programa a las distintas subpáginas, así como habilitar el enlace al programa en

la página índice del sitio web del Proyecto.

El programa también se enlaza desde ~/website/content/en/releng/_index.adoc.

Aproximadamente un mes antes del "code slush" planificado, el FreeBSD Release Engineering Team envía un correo electrónico recordatorio a los Desarrolladores de FreeBSD.

3. Terminología de la Ingeniería de Versiones

Esta sección describe parte de la terminología utilizada en el resto de este documento.

3.1. El Code Slush

Aunque el code slush no es una congelación completa del árbol, el FreeBSD Release Engineering Team solicita que los fallos existentes en el código base tengan prioridad sobre las nuevas características.

El code slush no impone aprobaciones en los commits a la rama.

3.2. La Congelación del Código

La congelación del código marca el punto en el tiempo en el que todos los commits a la rama requieren una aprobación explícita por parte del FreeBSD Release Engineering Team.

El repositorio de Subversion de FreeBSD tiene varios hooks para realizar comprobaciones de integridad antes de que se haga cualquier commit al árbol del código fuente. Uno de esos hooks evaluará si realizar el commit a una rama en particular requiere de aprobación específica.

Para forzar la aprobación de los commits por parte del FreeBSD Release Engineering Team, el Ingeniero de Lanzamiento actualiza base/svnadmin/conf/approvers y lo escribe en el repositorio. Una vez hecho esto, cualquier cambio en la rama debe incluir una línea "Approved by:" en el mensaje de commit.

La línea "Approved by:" debe coincidir con la segunda columna del archivo base/svnadmin/conf/approvers, de lo contrario, el commit será rechazado por los hooks del repositorio.

> Durante la congelación de código, se insta a los committers de FreeBSD a seguir las Change Request Guidelines.

3.3. La Congelación del KBI/KPI

La estabilidad de KBI/KPI implica que llamar a una función a través de dos versiones de software diferentes den el mismo resultado. Quien llama, ya sea un proceso, un subproceso o una función, espera que la función se comporte de cierta forma, de lo contrario, la estabilidad de KBI/KPI en la

rama se ve afectada.

4. Cambios en el Sitio Web Durante el Ciclo de Liberación

Esta sección describe los cambios que deberían de ocurrir en el sitio web a medida que avanza el ciclo de lanzamiento.

> Los ficheros especificados en esta sección son relativos a la rama `head/` del repositorio `doc` en Subversion.

4.1. Cambios en el Sitio Web Antes de que Empiece el Ciclo de Lanzamiento

Cuando el programa del ciclo de lanzamientos está disponible, estos archivos deben actualizarse para habilitar varias funcionalidades diferentes en la web del Proyecto FreeBSD:

Fichero a Editar	Qué Cambiar
~/shared/releases.adoc	Cambia `beta-upcoming` de `IGNORE` a `INCLUDE`
~/shared/releases.adoc	Cambia `beta-testing` de `IGNORE` a `INCLUDE`

4.2. Cambios del Sitio Web Durante `BETA` o `RC`

Cuando se transita de `PRERELEASE` a `BETA`, estos ficheros necesitan ser actualizados para habilitar el bloque "Help Test" en la página de descargas. Todos los ficheros son relativos a head/ en el repositorio `doc`:

Fichero a Editar	Qué Cambiar
share/releases.adoc	Actualiza `betarel-vers` a `BETA1`
~/website/data/en/news/news.toml	Añade una entrada anunciando la `BETA`
~/website/static/security/advisory-template.txt	Añade la nueva `BETA`, `RC` o `RELEASE` final a la plantilla
~/website/static/security/errata-template.txt	Añade la nueva `BETA`, `RC` o `RELEASE` final a la plantilla

Una vez que se crea la rama releng/12.0/, se necesita generar los distintos documentos relativos a la versión y añadirlos manualmente al repositorio `doc/`.

En release/doc, invoca para generar las páginas errata.html, hardware.html, readme.html, y relnotes.html, las cuales son añadidas luego a doc/head/en_US.ISO8859-1/htdocs/releases/X.YR/, donde *X.Y* representan los números de versión mayor y menor del lanzamiento.

Se debe establecer la propiedad `fbsd:nokeywords` en los nuevos ficheros añadidos para que los hooks

de pre-commit los añadan al repositorio.

Los documentos relevantes relacionados con una versión están en el repositorio doc para FreeBSD 12.x y posteriores.

4.3. Cambios en los Ports durante BETA, RC, y la RELEASE Final

Para cada construcción durante el ciclo de lanzamiento, los ficheros MANIFEST que contienen el SHA256 de los distintos conjuntos de distribuciones, como base.txz, kernel.txz y demás son añadidos al port misc/freebsd-release-manifests. Esto permite a otras utilidades, como ports-mgmt/poudriere, utilizar estos conjuntos de distribuciones de forma segura al proporcionar un mecanismo mediante el cual verificar los checksums.

5. Versiones desde head/

Esta sección describe los procedimientos generales del ciclo de lanzamiento de FreeBSD desde la rama head/.

5.1. Construcciones FreeBSD “ALPHA”

El concepto de construcción “ALPHA” se introdujo en el ciclo de FreeBSD 10.0-RELASE. A diferencia de las construcciones BETA y RC, las construcciones ALPHA no están incluidas en la programación de lanzamientos de FreeBSD.

La idea detrás de las construcciones ALPHA es proporcionar construcciones regulares de FreeBSD antes de la creación de la rama stable/.

Las instantáneas FreeBSD ALPHA se deberían construir aproximadamente una vez a la semana.

Para la primera construcción ALPHA, el valor BRANCH en sys/conf/newvers.sh tiene que cambiarse de CURRENT a ALPHA1. Para las siguientes construcciones ALPHA, incrementa cada valor ALPHAN en uno.

Visita Construir los Medios de Instalación de FreeBSD para obtener información acerca de cómo construir imágenes ALPHA.

5.2. Creando la Rama stable/12/

A la hora de crear la rama stable/ , se necesitan varios cambios tanto en la nueva rama stable/12/ como en la rama head/ . Los ficheros indicados son relativos a la raíz del repositorio. Para crear una nueva rama stable/12/ en Subversion:

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

Una vez que la rama stable/12/ ha sido escrita en el repositorio, haz los siguientes cambios:

Fichero a Editar	Qué Cambiar
stable/12/UPDATING	Actualiza la versión de FreeBSD y elimina el aviso acerca de `WITNESS`

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

stable/12/lib/clang/llvm.build.mk	Descomenta `-DNDEBUG`
stable/12/sys/*/conf/GENERIC*	Elimina el soporte para depuración
stable/12/sys/*/conf/MINIMAL	Elimina el soporte para depuración
stable/12/release/release.conf.sample	Actualiza `SRCBRANCH`
stable/12/sys/*/conf/GENERIC-NODEBUG	Elimina estas configuraciones del kernel
stable/12/sys/arm/conf/std.arm*	Elimina las opciones de depuración
stable/12/sys/conf/newvers.sh	Actualiza el valor de `BRANCH` a `BETA1`
stable/12/share/mk/src.opts.mk	Mueve `REPRODUCIBLE_BUILD` de `__DEFAULT_NO_OPTIONS` a `__DEFAULT_YES_OPTIONS`
stable/12/share/mk/src.opts.mk	Mueve `LLVM_ASSERTIONS` de `__DEFAULT_YES_OPTIONS` a `__DEFAULT_NO_OPTIONS` (sólo FreeBSD 13.x y posteriores)
stable/12/libexec/rc/rc.conf	Cambia `dumpdev` de `AUTO` a `NO` (es configurable para aquellos que lo quieren activado por defecto)
stable/12/release/Makefile	Elimina las entradas `debug.witness.trace`

Después, en la rama head/ , que se convertirá en una nueva versión mayor:

Fichero a Editar	Qué Cambiar
head/UPDATING	Actualiza la versión de FreeBSD
head/sys/conf/newvers.sh	Actualiza el valor de `BRANCH` para que muestre el valor `CURRENT` e incrementa `REVISION`
head/Makefile.inc1	Actualiza `TARGET_TRIPLE` y `MACHINE_TRIPLE`
head/sys/sys/param.h	Actualiza `__FreeBSD_version`
head/gnu/usr.bin/cc/cc_tools/freebsd-native.h	Actualiza `FBSD_MAJOR` y `FBSD_CC_VER`
head/contrib/gcc/config.gcc	Añade la sección `freebsdversion.h`
head/lib/clang/llvm.build.mk	Actualiza el valor de `OS_VERSION`
head/lib/clang/freebsd_cc_version.h	Actualiza `FREEBSD_CC_VERSION`
head/lib/clang/include/lld/Common/Version.inc	Actualiza `LLD_REVISION_STRING`
head/Makefile.libcompat	Actualiza `LIB32CPUFLAGS`

6. Versiones desde stable/

Esta sección describe los procedimientos generales del ciclo de lanzamiento de FreeBSD desde una rama stable/ establecida.

6.1. Code Sluch de la Rama FreeBSD `stable`

En preparación para la congelación del código en una rama `estable`, varios archivos deben ser actualizados para reflejar que el ciclo de liberación está oficialmente en curso. Estos archivos son todos relativos al nivel más alto de la rama estable:

Fichero a Editar	Qué Cambiar
sys/conf/newvers.sh	Actualiza el valor de `BRANCH` para que muestre `PRERELEASE`
Makefile.inc1	Actualiza `TARGET_TRIPLE`
lib/clang/llvm.build.mk	Actualiza `OS_VERSION`
Makefile.libcompat	Actualiza `LIB32CPUFLAGS`

6.2. Construcciones FreeBSD `BETA`

Después del code slush, la siguiente fase del ciclo de lanzamiento es la congelación del código. Este es el punto en el que todos los commits a la rama estable requieren de una autorización explícita del FreeBSD Release Engineering Team. Esto se fuerza mediante unos hook de pre-commit en el repositorio de Subversion, editando base/svnadmin/conf/approvers para incluir una expresión regular que concuerde con la rama stable/12/ para la versión:

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

> Hay dos excepciones, las cuales no requieren la aprobación del commit. La primera es cualquier cambio que deba realizar el Ingeniero de Lanzamientos para continuar con el flujo de trabajo diario del ciclo de lanzamientos, y el segundo, son las correcciones de errores de seguridad que puedan ocurrir durante el ciclo de lanzamientos.

Una vez que la congelación de código tiene efecto, la siguiente construcción de la rama se etiqueta como `BETA1`. Esto se hace actualizando el valor `BRANCH` en sys/conf/newvers.sh de `PRERELEASE` a `BETA1`.

Una vez hecho esto, se comienza la construcción del primer conjunto `BETA`. Las construcciones `BETA` subsiguientes no requieren más actualización que la del fichero sys/conf/newvers.sh, incrementando el número de construcción `BETA`.

6.3. Creando la Rama releng/12.0/

Cuando la primera construcción RC (Release Candidate) está lista para comenzar, se crea la rama releng/12.0/. Este es un proceso de varios pasos que se tiene que realizar en un orden específico, para evitar anomalías como el solapado de valores de __FreeBSD_version, por ejemplo. Las rutas mostradas abajo son relativas a la raíz del repositorio. El orden de los commits y lo que hay que cambiar es el siguiente:

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

Fichero a Editar	Qué Cambiar
releng/12.0/sys/conf/newvers.sh	Cambia BETAX a RC1
releng/12.0/sys/sys/param.h	Actualiza __FreeBSD_version
[.filename]#releng/12.0/sys/conf/kern.opts.mk	Cambia REPRODUCIBLE_BUILD de DEFAULT_NO_OPTIONS aDEFAULT_YES_OPTIONS
releng/12.0/etc/pkg/FreeBSD.conf	Sustituye latest con quarterly para la localización por defecto del repositorio de paquetes
releng/12.0/release/pkg_repos/release-dvd.conf	Sustituye latest con quarterly para la localización por defecto del repositorio de paquetes
stable/12/sys/conf/newvers.sh	Actualiza BETAX con PRERELEASE
stable/12/sys/sys/param.h	Actualiza __FreeBSD_version
svnadmin/conf/approvers	Añade una nueva línea de validadores a la rama releng del mismo modo que se hizo para la rama stable

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

Ahora que ya existe el nuevo valor de __FreeBSD_version, actualiza también ~/documentation/content/en/books/porters-handbook/versions/chapter.adoc en el repositorio del Proyecto de Documentación.

Después de que la primera construcción RC se haya completado y haya sido probada, la rama stable/ puede ser "descongelada" quitando (o comentando) la entrada ^/stable/12/ en svnadmin/conf/approvers.

Cuando la primera RC esté disponible, se debería enviar un correo a FreeBSD Bugmeister Team para añadir la nueva -RELEASE de FreeBSD a las versiones disponibles en el menú desplegable que se muestra en el gestor de bugs.

7. Construir los Medios de Instalación de FreeBSD

Esta sección describe los procedimientos generales que producen instantáneas y versiones del desarrollo de FreeBSD.

7.1. Scripts de Construcción de Versiones

Esta sección describe los scripts de construcción utilizados por FreeBSD Release Engineering Team para producir instantáneas de desarrollo y versiones.

7.1.1. El Script release.sh

Antes de la versión FreeBSD 9.0-RELEASE, se actualizó src/release/Makefile para soportar, y el script src/release/generate-release.sh fue introducido como un envoltorio, para automatizar la invocación de los distintos objetivos.

Antes de FreeBSD 9.2-RELEASE, se introdujo src/release/release.sh, que basado en gran medida en src/release/generate-release.sh incluía soporte para especificar archivos de configuración para anular varias opciones y variables de entorno. El soporte para los archivos de configuración proporcionaba soporte para construir de forma cruzada cada arquitectura para una versión especificando un archivo de configuración separado para cada invocación.

Un pequeño ejemplo de cómo usar src/release/release.sh para construir una única versión en /scratch:

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

Un breve ejemplo del uso de src/release/release.sh para construir una única versión cruzada utilizando un directorio de destino diferente, cree un release.conf personalizado que contenga:

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

Después invoca src/release/release.sh así:

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

Echa un vistazo a src/release/release.conf.sample para más detalles y ejemplos de uso.

7.1.2. El Script Envoltorio thermite.sh

Para que la construcción cruzada del conjunto completo de arquitecturas soportadas en una rama determinada sea más rápida y fácil, y para reducir los factores de error humano, se escribió un script de envoltura alrededor de src/release/release.sh para iterar a través de las diversas combinaciones de arquitecturas e invocar src/release/release.sh utilizando un archivo de configuración específico para esa arquitectura.

El script de envoltura se llama thermite.sh, el cual está disponible en el repositorio Subversion de FreeBSD en `svn://svn.freebsd.org/base/user/gjb/thermite/`, junto con ficheros de configuración para construir las instantáneas de desarrollo de head/ y stable/12/.

El uso de thermite.sh se trata en Construyendo Instantáneas de Desarrollo de FreeBSD y Construyendo Versiones de FreeBSD.

Cada arquitectura y núcleo individual tiene su propio archivo de configuración usado por release.sh. Cada rama tiene su propia configuración defaults-X.conf, que contiene entradas comunes en cada arquitectura, donde se establecen y/o anulan las anulaciones o variables especiales en los archivos por construcción.

El esquema para nombrar los ficheros de configuración de cada construcción es de la forma ${revision}-${TARGET_ARCH}-${KERNCONF}-${type}.conf, donde las variables en mayúsculas son equivalentes a lo que se utiliza en el sistema de construcción y las variables en minúsculas se establecen dentro de los ficheros de configuración, mapeando al número de versión mayor de la respectiva rama.

Cada rama también tiene su propia configuración builds-X.conf, que es usada por thermite.sh. El script thermite.sh itera a través de cada valor ${revision}, ${TARGET_ARCH}, ${KERNCONF}, y ${type}, creando una lista maestra de lo que hay que construir. Sin embargo, una determinada combinación de la lista sólo se construirá si existe el respectivo archivo de configuración, que es donde el esquema de denominación anterior es relevante.

Hay dos caminos para la obtención de archivos:

- builds-12.conf - main.conf

  Este controla el comportamiento de thermite.sh

- 12-amd64-GENERIC-snap.conf - defaults-12.conf - main.conf

  Este controla el comportamiento de release/release.sh en la construcción

> Los ficheros de configuración builds-12.conf, defaults-12.conf, y main.conf existen para reducir la repetición entre los archivos de cada construcción.

7.2. Construyendo Instantáneas de Desarrollo de FreeBSD

Las máquinas de construcción de la versión oficial tienen un diseño de sistema de archivos

específico, que usando ZFS, thermite.sh se aprovecha mucho con los clones y las instantáneas, asegurando un ambiente de construcción prístino.

El script de construcción se encuentra en /releng/scripts-snapshot/scripts o /releng/scripts-release/scripts respectivamente, para evitar colisiones entre una construcción `RC` de una rama releng y una instantánea `STABLE`de la rama estable respectiva.

Existe un conjunto separado de datos para las construcciones de las imágenes finales, /snap/ftp. Este directorio contiene los directorios tanto de las instantáneas como de las versiones. Sólo se usan si la variable `EVERYTHINGISFINE` está definida en main.conf.

> El nombre de la variable `EVERYTHINGISFINE` se escogió para evitar colisiones con una variable que pudiera estar establecida en el entorno del usuario, habilitando accidentalmente el comportamiento que depende de ella al estar definida.

Conforme thermite.sh itera por la lista maestra de combinaciones y localiza el fichero de configuración de cada construcción, se crea un conjunto de datos ZFS bajo /releng, tales como /releng/12-amd64-GENERIC-snap. Los árboles `src/`, `ports/`, y `doc/`se descargan en conjuntos de datos ZFS separados, tales como /releng/12-src-snap, los cuales son clonados después y montados en sus respectivos conjuntos de datos. Esto se hace para evitar descargar un árbol más de una vez.

Asumiendo estas rutas de sistemas de archivos, thermite.sh sería invocado como:

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

Una vez que se han completado las compilaciones, se dispone de scripts auxiliares adicionales para generar correos electrónicos de instantáneas de desarrollo que se envían a la lista de correo `freebsd-snapshots@freebsd.org`:

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

> La salida generada debe ser verificada dos veces para comprobar que es correcta, y el correo electrónico en sí debe ser firmado en línea mediante PGP.

> Los scripts de apoyo sólo se utilizan en las construcciones de instantáneas de desarrollo. Los anuncios durante un ciclo de liberación (excepto el anuncio de la versión definitiva) se crean a partir de una plantilla de correo. Un ejemplo de la plantilla de correo que se usa actualmente se puede encontrar aquí.

7.3. Construyendo Versiones de FreeBSD

De forma similar para la construcción de instantáneas de desarrollo de FreeBSD, thermite.sh se

invocaría del mismo modo. La diferencia entre instantáneas de desarrollo y construcciones de versión, `BETA` y `RC` incluidas, es que los ficheros de configuración deben utilizar el tipo `release` en lugar de `snap` como se mencionaba anteriormente.

Además, `BUILDTYPE` y `types` se deben cambiar de `snap` a `release` en defaults-12.conf y builds-12.conf, respectivamente.

Cuando se construya `BETA`, `RC` y la `RELEASE` final, también hay que establecer estáticamente `BUILDSVNREV` a la revisión de la rama que refleja el cambio de nombre, `BUILDDATE` a la fecha en la que las construcciones comienzan en formato `YYYYMMDD`. Si los árboles `doc/` y `ports/` han sido etiquetados, hay que establecer también `PORTBRANCH` y `DOCBRANCH` a las rutas de las etiquetas relevantes en el repositorio de Subversion, reemplazando `HEAD` con la última revisión. También hay que establecer `releasesrc` en builds-12.conf a la rama relevante, como stable/12/ o releng/12.0/.

Durante el ciclo de lanzamiento, se almacena una copia de CHECKSUM.SHA512 y CHECKSUM.SHA256 para cada arquitectura en el repositorio interno de FreeBSD Release Engineering Team además de ser incluido en varios correos electrónicos de anuncio. Cada MANIFEST que contiene los hashes de base.txz, kernel.txz, etc. es añadido también a misc/freebsd-release-manifests en la Colección de Ports.

En preparación para la construcción de la liberación, varios archivos necesitan ser actualizados:

Fichero a Editar	Qué Cambiar
sys/conf/newvers.sh	Actualiza el valor de `BRANCH` a `RELEASE`
UPDATING	Añade la fecha prevista del anuncio
lib/csu/common/crtbrand.S	Reemplaza `__FreeBSD_version` con el valor del fichero sys/sys/param.h

Después de construir la `RELEASE` final, la rama releng/12.0/ se etiqueta como releng/12.0/ utilizando la revisión a partir de la cual se construyó la `RELEASE`. De forma similar a la creación de las ramas stable/12/ y releng/12.0/, esto se hace con `svn cp`. Desde la raíz del repositorio:

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

8. Publicar los Medios de Instalación de FreeBSD para los Mirrors del Proyecto

Esta sección describe el procedimiento para publicar instantáneas del desarrollo de FreeBSD y las versiones en los mirrors del Proyecto.

8.1. Puesta en marcha las Imágenes de los Medios de Instalación de FreeBSD

La puesta en marcha de las instantáneas y lanzamientos de FreeBSD es un proceso de dos partes:

- Crear la estructura de directorios que concuerda con la jerarquía en `ftp-master`

  Si `EVERYTHINGISFINE` está definida en los ficheros de configuración de la construcción, main.conf en el caso de los scripts de construcción referenciados arriba, esto sucede automáticamente después de que se haya completado la construcción, creando la estructura de directorios en ${DESTDIR}/R/ftp-stage con una estructura de rutas que concuerda con lo que se espera en `ftp-master`. Esto es equivalente a ejecutar directamente lo siguiente:

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

  Después de que se haya construido cada arquitectura, thermite.sh sincronizará con rsync el ${DESTDIR}/R/ftp-stage de la construcción a /snap/ftp/snapshots o /snap/ftp/releases en la máquina de construcción, respectivamente.

- Copiar los ficheros a un directorio de preparación en `ftp-master` antes de mover los ficheros a pub/ para comenzar la propagación a los mirrors del Proyecto

  Una vez que todas las construcciones han terminado, /snap/ftp/snapshots, o /snap/ftp/releases para una release, es consultado por `ftp-master` utilizando rsync para /archive/tmp/snapshots o /archive/tmp/releases, respectivamente.

> En `ftp-master` en la infraestructura del Proyecto FreeBSD, este proceso necesita nivel de acceso `root`, ya que este paso debe ejecutarse como el usuario `archive`.

8.2. Publicación de los Medios de Instalación de FreeBSD

Una vez que las imágenes están preparadas en /archive/tmp/, están listas para ser publicadas poniéndolas en /archive/pub/FreeBSD. Para reducir el tiempo de propagación se crean enlaces desde /archive/tmp a /archive/pub/FreeBSD.

> Para que esto sea efectivo, tanto /archive/tmp como /archive/pub deben estar en el mismo sistema de ficheros lógico.

Hay un problema, sin embargo, donde se debe usar rsync después para corregir los enlaces simbólicos en pub/FreeBSD/snapshots/ISO-IMAGES que serán reemplazados con enlaces duros, incrementando el tiempo de propagación.

> Al igual que con los pasos de preparación, este requiere acceso nivel `root` ya que este paso debe ser ejecutado como el usuario `archive`.

Como usuario `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/
```

Reemplaza *snapshots* con *releases* donde corresponda.

9. Cerrando el Ciclo de Liberación

En esta sección se describen las tareas generales posteriores a la liberación.

9.1. Notificaciones de Errores Posteriores a la Liberación

Según se acerca el final del ciclo de liberación, es común tener varios EN candidatos (Errata Notice) para tratar problemas que han sido descubiertos tarde en el ciclo. Después de la release, el FreeBSD Release Engineering Team y el FreeBSD Security Team revisan cambios que no fueron aprobados antes de la release final, y dependiendo del alcance del cambio en cuestión, podría crear una EN.

> El proceso de creación de ENs es manejado por el FreeBSD Security Team.

Para solicitar una Errata Notice después de que el ciclo de liberación haya finalizado, un desarrollador debería rellenar el Errata Notice template, en concreto las secciones `Background`, `Problem Description`, `Impact`, y `Workaround` si es aplicable.

La plantilla completa de la Errata Notice se debería enviar junto con un parche contra releng/ o una lista de revisiones de la rama stable/.

Para peticiones de Errata Notice que sean inmediatamente posteriores a la release, la petición debería ser enviada por correo electrónico al FreeBSD Release Engineering Team y al FreeBSD Security Team. Una vez que la rama releng/ ha sido entregada al FreeBSD Security Team como se describe en Entrega al FreeBSD Security Team, las peticiones de Errata Notice deberían ser enviadas al FreeBSD Security Team.

9.2. Entrega al FreeBSD Security Team

Aproximadamente dos semanas después de la release, el Ingeniero de Liberación actualiza svnadmin/conf/approvers cambiando la columna `approver` de `re` a `(so|security-officer)` para la rama releng/12.0/.

10. Fin del ciclo de Vida de la Versión

Esta sección describe los ficheros que hay que actualizar en el sitio web cuando una versión alcanza su EoL (End-of-Life).

10.1. Actualizaciones del Sitio Web para End-Of-Life

Cuando un lanzamiento llega al final de su vida, las referencias a ese lanzamiento deben ser eliminadas y/o actualizadas en el sitio web:

Fichero	Qué Cambiar
~/website/themes/beastie/layouts/index.html	Elimina las referencias a `u-relXXX-announce` y `u-relXXX-announce`.
~/website/content/en/releases/_index.adoc	Mueve las variables `u-relXXX-*` desde la lista de versiones soportadas a la lista de versiones obsoletas (Legacy Releases).
~/website/content/en/releng/_index.adoc	Actualiza la rama releng apropiada para que se refleje que la rama ya no está soportada.
~/website/content/en/security/_index.adoc	Elimina la rama de la lista de ramas soportadas.
~/website/content/en/where.adoc	Elimina las URLs para la release.
~/website/themes/beastie/layouts/partials/sidenav.html	Elimina las referencias a `u-relXXX-announce` y `u-relXXX-announce`.
~/website/static/security/advisory-template.txt	Elimina las referencias a las ramas de release y releng.
~/website/static/security/errata-template.txt	Elimina las referencias a las ramas de release y releng.