FreeBSD Release Engineering

Abstract

This document is outdated and does not accurately describe the current release procedures of the FreeBSD Release Engineering team. It is retained for historical purposes. The current procedures used by the FreeBSD Release Engineering team are available in the FreeBSD Release Engineering article.

This paper describes the approach used by the FreeBSD release engineering team to make production quality releases of the FreeBSD Operating System. It details the methodology used for the official FreeBSD releases and describes the tools available for those interested in producing customized FreeBSD releases for corporate rollouts or commercial productization.

Table of Contents

1. Introduction . . . 1
2. Release Process . . . 3
3. Release Building . . . 8
4. Distribution . . . 10
5. Extensibility . . . 11
6. Lessons Learned from FreeBSD 4.4 . . . 11
7. Future Directions . . . 12
8. Acknowledgements . . . 12

1. Introduction

The development of FreeBSD is a very open process. FreeBSD is comprised of contributions from thousands of people around the world. The FreeBSD Project provides Subversion [1] access to the general public so that others can have access to log messages, diffs (patches) between development branches, and other productivity enhancements that formal source code management provides. This has been a huge help in attracting more talented developers to FreeBSD. However, I think everyone would agree that chaos would soon manifest if write access to the main repository was opened up to everyone on the Internet. Therefore only a "select" group of nearly 300 people are given write access to the Subversion repository. These FreeBSD committers[2] are usually the people who do the bulk of FreeBSD development. An elected Core Team[3] of developers provide some level of direction over the project.

The rapid pace of `FreeBSD` development makes the main development branch unsuitable for the everyday use by the general public. In particular, stabilizing efforts are required for polishing the development system into a production quality release. To solve this conflict, development continues on several parallel tracks. The main development branch is the *HEAD* or *trunk* of our Subversion

tree, known as "FreeBSD-CURRENT" or "-CURRENT" for short.

A set of more stable branches are maintained, known as "FreeBSD-STABLE" or "-STABLE" for short. All branches live in a master Subversion repository maintained by the FreeBSD Project. FreeBSD-CURRENT is the "bleeding-edge" of FreeBSD development where all new changes first enter the system. FreeBSD-STABLE is the development branch from which major releases are made. Changes go into this branch at a different pace, and with the general assumption that they have first gone into FreeBSD-CURRENT and have been thoroughly tested by our user community.

The term *stable* in the name of the branch refers to the presumed Application Binary Interface stability, which is promised by the project. This means that a user application compiled on an older version of the system from the same branch works on a newer system from the same branch. The ABI stability has improved greatly from the compared to previous releases. In most cases, binaries from the older *STABLE* systems run unmodified on newer systems, including *HEAD*, assuming that the system management interfaces are not used.

In the interim period between releases, weekly snapshots are built automatically by the FreeBSD Project build machines and made available for download from `https:/download.FreeBSD.org/snapshots/`. The widespread availability of binary release snapshots, and the tendency of our user community to keep up with -STABLE development with Subversion and “make buildworld” [4] helps to keep FreeBSD-STABLE in a very reliable condition even before the quality assurance activities ramp up pending a major release.

In addition to installation ISO snapshots, weekly virtual machine images are also provided for use with VirtualBox, qemu, or other popular emulation software. The virtual machine images can be downloaded from `https://download.FreeBSD.org/snapshots/VM-IMAGES/`.

The virtual machine images are approximately 150MB xz(1) compressed, and contain a 10GB sparse filesystem when attached to a virtual machine.

Bug reports and feature requests are continuously submitted by users throughout the release cycle. Problems reports are entered into our Bugzilla database through the web interface provided at https://www.freebsd.org/support/bugreports/.

To service our most conservative users, individual release branches were introduced with FreeBSD 4.3. These release branches are created shortly before a final release is made. After the release goes out, only the most critical security fixes and additions are merged onto the release branch. In addition to source updates via Subversion, binary patchkits are available to keep systems on the *releng/X.Y* branches updated.

1.1. What This Article Describes

The following sections of this article describe:

Release Process

The different phases of the release engineering process leading up to the actual system build.

Release Building

The actual build process.

Extensibility

How the base release may be extended by third parties.

Lessons Learned from FreeBSD 4.4

Some of the lessons learned through the release of FreeBSD 4.4.

Future Directions

Future directions of development.

2. Release Process

New releases of FreeBSD are released from the -STABLE branch at approximately four month intervals. The FreeBSD release process begins to ramp up 70-80 days before the anticipated release date when the release engineer sends an email to the development mailing lists to remind developers that they only have 15 days to integrate new changes before the code freeze. During this time, many developers perform what have become known as "MFC sweeps".

MFC stands for "Merge From CURRENT" and it describes the process of merging a tested change from our -CURRENT development branch to our -STABLE branch. Project policy requires any change to be first applied to trunk, and merged to the -STABLE branches after sufficient external testing was done by -CURRENT users (developers are expected to extensively test the change before committing to -CURRENT, but it is impossible for a person to exercise all usages of the general-purpose operating system). Minimal MFC period is 3 days, which is typically used only for trivial or critical bugfixes.

2.1. Code Review

Sixty days before the anticipated release, the source repository enters a "code freeze". During this time, all commits to the -STABLE branch must be approved by `Release Engineering Team <re@FreeBSD.org>`. The approval process is technically enforced by a pre-commit hook. The kinds of changes that are allowed during this period include:

- Bug fixes.
- Documentation updates.
- Security-related fixes of any kind.
- Minor changes to device drivers, such as adding new Device IDs.
- Driver updates from the vendors.
- Any additional change that the release engineering team feels is justified, given the potential risk.

Shortly after the code freeze is started, a *BETA1* image is built and released for widespread testing. During the code freeze, at least one beta image or release candidate is released every two weeks until the final release is ready. During the days preceding the final release, the release engineering team is in constant communication with the security-officer team, the documentation maintainers, and the port maintainers to ensure that all of the different components required for a successful

release are available.

After the quality of the BETA images is satisfying enough, and no large and potentially risky changes are planned, the release branch is created and *Release Candidate* (RC) images are built from the release branch, instead of the BETA images from the STABLE branch. Also, the freeze on the STABLE branch is lifted and release branch enters a "hard code freeze" where it becomes much harder to justify new changes to the system unless a serious bug-fix or security issue is involved.

2.2. Final Release Checklist

When several BETA images have been made available for widespread testing and all major issues have been resolved, the final release "polishing" can begin.

2.2.1. Creating the Release Branch

In all examples below, `$FSVN` refers to the location of the FreeBSD Subversion repository, `svn+ssh://svn.FreeBSD.org/base/`.

The layout of FreeBSD branches in Subversion is described in the Committer's Guide. The first step in creating a branch is to identify the revision of the `stable/X` sources that you want to branch *from*.

```
# svn log -v $FSVN/stable/9
```

The next step is to create the *release branch*

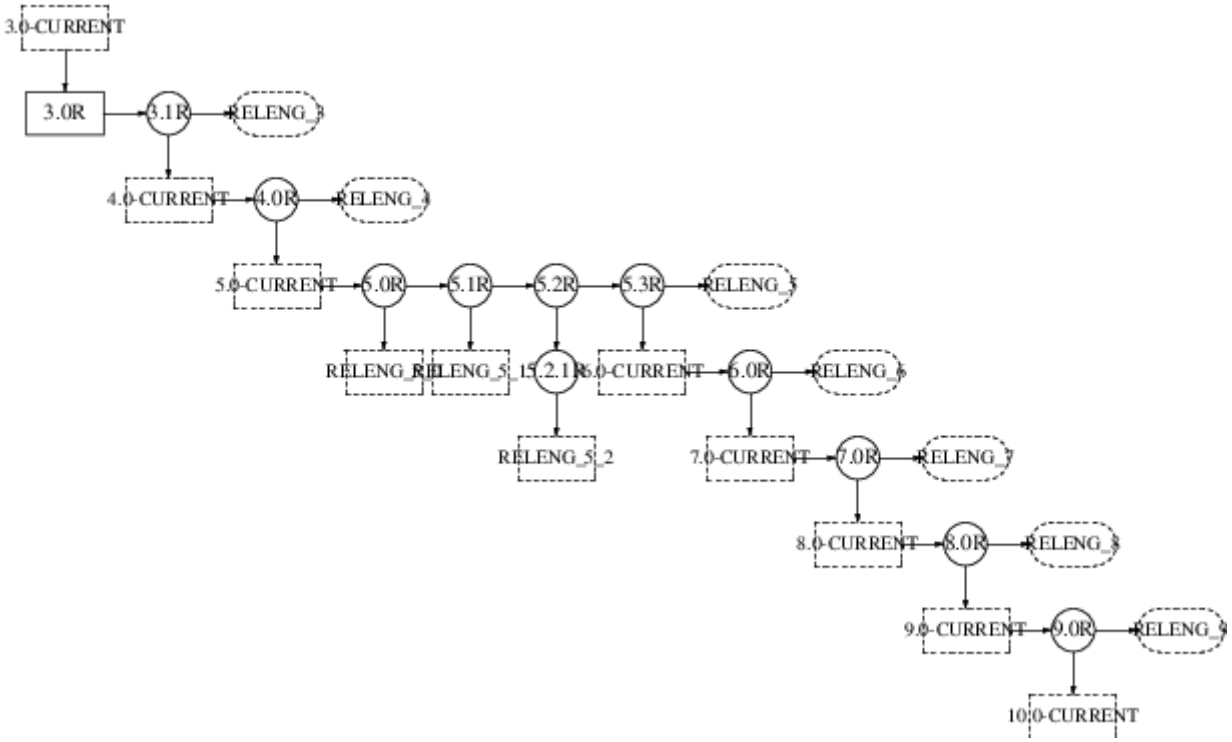
```
# svn cp $FSVN/stable/9@REVISION $FSVN/releng/9.2
```

This branch can be checked out:

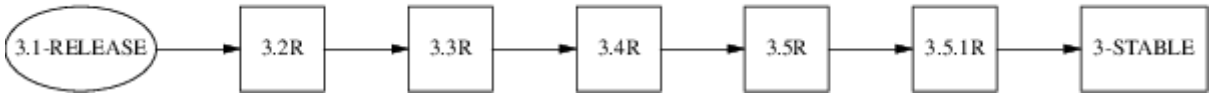
```
# svn co $FSVN/releng/9.2 src
```

Creating the `releng` branch and `release` tags is done by the Release Engineering Team.

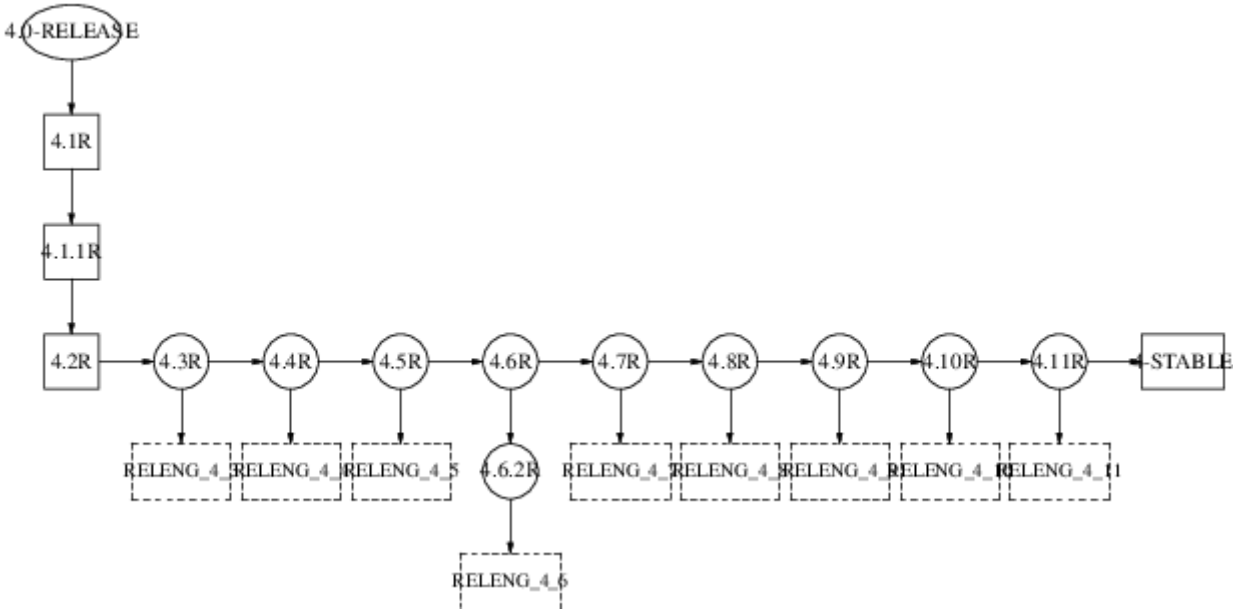
H E A D (Main Development Branch)

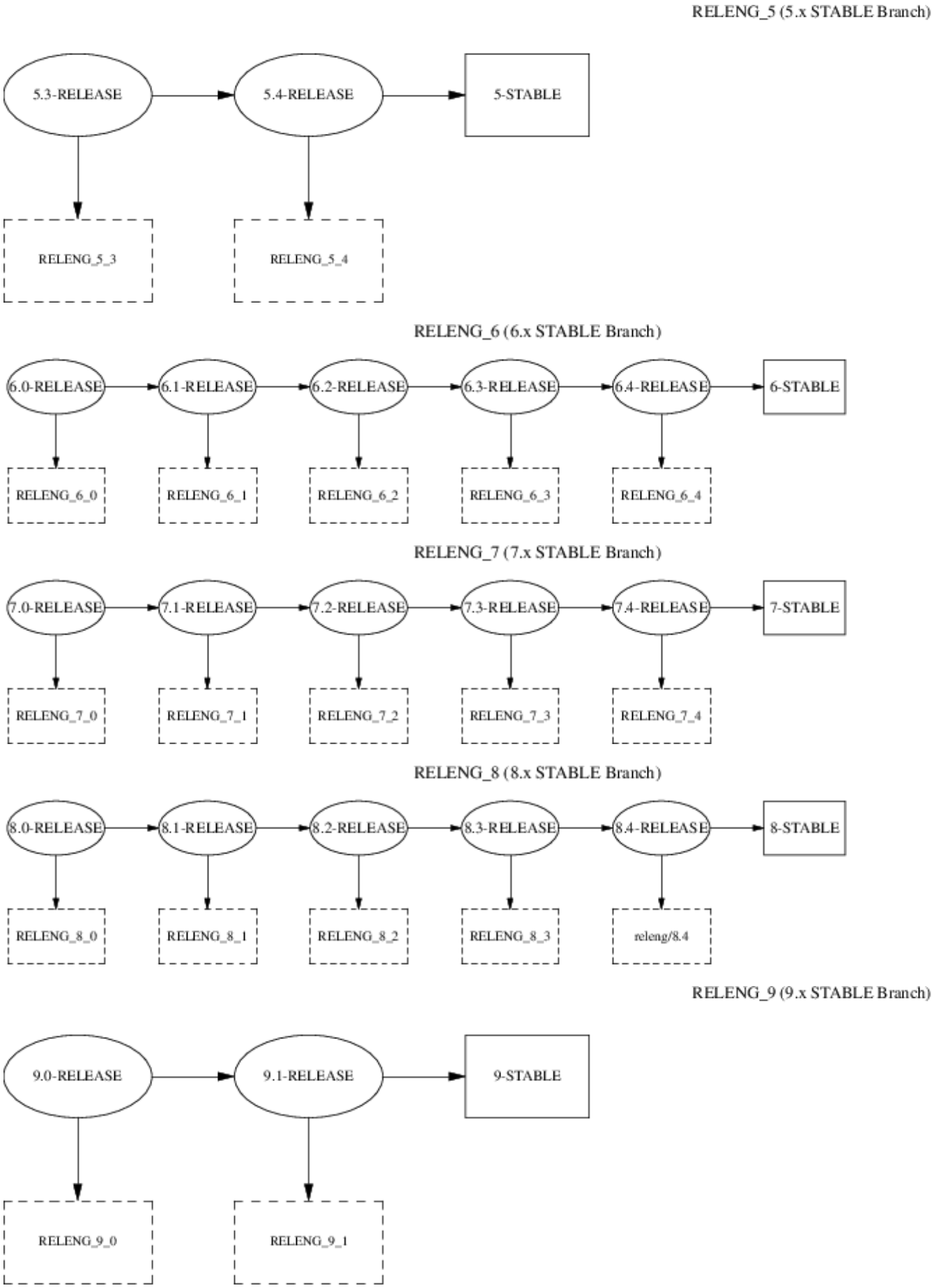


RELENG_3 (3.x STABLE Branch)



RELENG_4 (4.x STABLE Branch)





2.2.2. Bumping up the Version Number

Before the final release can be tagged, built, and released, the following files need to be modified to

reflect the correct version of FreeBSD:

- doc/en_US.ISO8859-1/books/handbook/mirrors/chapter.xml
- doc/en_US.ISO8859-1/books/porters-handbook/book.xml
- doc/en_US.ISO8859-1/htdocs/cgi/ports.cgi
- ports/Tools/scripts/release/config
- doc/shared/xml/freebsd.ent
- src/Makefile.inc1
- src/UPDATING
- src/gnu/usr.bin/groff/tmac/mdoc.local
- src/release/Makefile
- src/release/doc/en_US.ISO8859-1/shared/xml/release.dsl
- src/release/doc/shared/examples/Makefile.relnotesng
- src/release/doc/shared/xml/release.ent
- src/sys/conf/newvers.sh
- src/sys/sys/param.h
- src/usr.sbin/pkg_install/add/main.c
- doc/en_US.ISO8859-1/htdocs/search/opensearch/man.xml

The release notes and errata files also need to be adjusted for the new release (on the release branch) and truncated appropriately (on the stable/current branch):

- src/release/doc/en_US.ISO8859-1/relnotes/common/new.xml
- src/release/doc/en_US.ISO8859-1/errata/article.xml

Sysinstall should be updated to note the number of available ports and the amount of disk space required for the Ports Collection. [5] This information is currently kept in src/usr.sbin/bsdinstall/dist.c.

After the release has been built, a number of files should be updated to announce the release to the world. These files are relative to `head/` within the `doc/` subversion tree.

- share/images/articles/releng/branches-relengX.pic
- head/shared/xml/release.ent
- en_US.ISO8859-1/htdocs/releases/*
- en_US.ISO8859-1/htdocs/releng/index.xml
- share/xml/news.xml

Additionally, update the "BSD Family Tree" file:

- src/shared/misc/bsd-family-tree

2.2.3. Creating the Release Tag

When the final release is ready, the following command will create the `release/9.2.0` tag.

```
# svn cp $FSVN/releng/9.2 $FSVN/release/9.2.0
```

The Documentation and Ports managers are responsible for tagging their respective trees with the `tags/RELEASE_9_2_0` tag.

When the Subversion `svn cp` command is used to create a *release tag*, this identifies the source at a specific point in time. By creating tags, we ensure that future release builders will always be able to use the same source we used to create the official FreeBSD Project releases.

3. Release Building

FreeBSD "releases" can be built by anyone with a fast machine and access to a source repository. (That should be everyone, since we offer Subversion access! See the Subversion section in the Handbook for details.) The *only* special requirement is that the md(4) device must be available. If the device is not loaded into your kernel, then the kernel module should be automatically loaded when mdconfig(8) is executed during the boot media creation phase. All of the tools necessary to build a release are available from the Subversion repository in src/release. These tools aim to provide a consistent way to build FreeBSD releases. A complete release can actually be built with only a single command, including the creation of ISO images suitable for burning to CDROM or DVD, and an FTP install directory. release(7) fully documents the `src/release/generate-release.sh` script which is used to build a release. `generate-release.sh` is a wrapper around the Makefile target: `make release`.

3.1. Building a Release

release(7) documents the exact commands required to build a FreeBSD release. The following sequences of commands can build an 9.2.0 release:

```
# cd /usr/src/release
# sh generate-release.sh release/9.2.0 /local3/release
```

After running these commands, all prepared release files are available in /local3/release/R directory.

The release Makefile can be broken down into several distinct steps.

- Creation of a sanitized system environment in a separate directory hierarchy with “make installworld”.
- Checkout from Subversion of a clean version of the system source, documentation, and ports into the release build hierarchy.
- Population of /etc and /dev in the chrooted environment.

- chroot into the release build hierarchy, to make it harder for the outside environment to taint this build.
- `make world` in the chrooted environment.
- Build of Kerberos-related binaries.
- Build GENERIC kernel.
- Creation of a staging directory tree where the binary distributions will be built and packaged.
- Build and installation of the documentation toolchain needed to convert the documentation source (SGML) into HTML and text documents that will accompany the release.
- Build and installation of the actual documentation (user manuals, tutorials, release notes, hardware compatibility lists, and so on.)
- Package up distribution tarballs of the binaries and sources.
- Create FTP installation hierarchy.
- *(optionally)* Create ISO images for CDROM/DVD media.

For more information about the release build infrastructure, please see release(7).

> It is important to remove any site-specific settings from /etc/make.conf. For example, it would be unwise to distribute binaries that were built on a system with `CPUTYPE` set to a specific processor.

3.2. Contributed Software ("ports")

The FreeBSD Ports collection is a collection of over 36000 third-party software packages available for FreeBSD. The `Ports Management Team <portmgr@FreeBSD.org>` is responsible for maintaining a consistent ports tree that can be used to create the binary packages that accompany official FreeBSD releases.

3.3. Release ISOs

Starting with FreeBSD 4.4, the FreeBSD Project decided to release all four ISO images that were previously sold on the *BSDi/Wind River Systems/FreeBSD Mall* "official" CDROM distributions. Each of the four discs must contain a README.TXT file that explains the contents of the disc, a CDROM.INF file that provides meta-data for the disc so that bsdinstall(8) can validate and use the contents, and a filename.txt file that provides a manifest for the disc. This *manifest* can be created with a simple command:

```
/stage/cdrom# find . -type f | sed -e 's/^\.\///' | sort > filename.txt
```

The specific requirements of each CD are outlined below.

3.3.1. Disc 1

The first disc is almost completely created by `make release`. The only changes that should be made

to the disc1 directory are the addition of a tools directory, and as many popular third party software packages as will fit on the disc. The tools directory contains software that allow users to create installation floppies from other operating systems. This disc should be made bootable so that users of modern PCs do not need to create installation floppy disks.

If a custom kernel of FreeBSD is to be included, then bsdinstall(8) and release(7) must be updated to include installation instructions. The relevant code is contained in src/release and src/usr.sbin/bsdinstall. Specifically, the file src/release/Makefile, and dist.c, dist.h, menus.c, install.c, and Makefile will need to be updated under src/usr.sbin/bsdinstall. Optionally, you may choose to update bsdinstall.8.

3.3.2. Disc 2

The second disc is also largely created by `make release`. This disc contains a "live filesystem" that can be used from bsdinstall(8) to troubleshoot a FreeBSD installation. This disc should be bootable and should also contain a compressed copy of the CVS repository in the CVSROOT directory and commercial software demos in the commerce directory.

3.3.3. Multi-volume Support

Sysinstall supports multiple volume package installations. This requires that each disc have an INDEX file containing all of the packages on all volumes of a set, along with an extra field that indicates which volume that particular package is on. Each volume in the set must also have the `CD_VOLUME` variable set in the cdrom.inf file so that bsdinstall can tell which volume is which. When a user attempts to install a package that is not on the current disc, bsdinstall will prompt the user to insert the appropriate one.

4. Distribution

4.1. FTP Sites

When the release has been thoroughly tested and packaged for distribution, the master FTP site must be updated. The official FreeBSD public FTP sites are all mirrors of a master server that is open only to other FTP sites. This site is known as `ftp-master`. When the release is ready, the following files must be modified on `ftp-master`:

/pub/FreeBSD/releases/arch/X.Y-RELEASE/

The installable FTP directory as output from `make release`.

/pub/FreeBSD/ports/arch/packages-X.Y-release/

The complete package build for this release.

/pub/FreeBSD/releases/arch/X.Y-RELEASE/tools

A symlink to ../../../tools.

/pub/FreeBSD/releases/arch/X.Y-RELEASE/packages

A symlink to ../../../ports/arch/packages-X.Y-release.

/pub/FreeBSD/releases/arch/ISO-IMAGES/X.Y/X.Y-RELEASE-arch-*.iso

The ISO images. The "*" is disc1, disc2, etc. Only if there is a disc1 and there is an alternative first installation CD (for example a stripped-down install with no windowing system) there may be a mini as well.

For more information about the distribution mirror architecture of the FreeBSD FTP sites, please see the Mirroring FreeBSD article.

It may take many hours to two days after updating `ftp-master` before a majority of the Tier-1 FTP sites have the new software depending on whether or not a package set got loaded at the same time. It is imperative that the release engineers coordinate with the FreeBSD mirror site administrators before announcing the general availability of new software on the FTP sites. Ideally the release package set should be loaded at least four days prior to release day. The release bits should be loaded between 24 and 48 hours before the planned release time with "other" file permissions turned off. This will allow the mirror sites to download it but the general public will not be able to download it from the mirror sites. Mail should be sent to FreeBSD mirror site administrators at the time the release bits get posted saying the release has been staged and giving the time that the mirror sites should begin allowing access. Be sure to include a time zone with the time, for example make it relative to GMT.

4.2. CD-ROM Replication

Coming soon: Tips for sending FreeBSD ISOs to a replicator and quality assurance measures to be taken.

5. Extensibility

Although FreeBSD forms a complete operating system, there is nothing that forces you to use the system exactly as we have packaged it up for distribution. We have tried to design the system to be as extensible as possible so that it can serve as a platform that other commercial products can be built on top of. The only "rule" we have about this is that if you are going to distribute FreeBSD with non-trivial changes, we encourage you to document your enhancements! The FreeBSD community can only help support users of the software we provide. We certainly encourage innovation in the form of advanced installation and administration tools, for example, but we cannot be expected to answer questions about it.

5.1. Scripting `bsdinstall`

The FreeBSD system installation and configuration tool, bsdinstall(8), can be scripted to provide automated installs for large sites. This functionality can be used in conjunction with Intel® PXE [6] to bootstrap systems from the network.

6. Lessons Learned from FreeBSD 4.4

The release engineering process for 4.4 formally began on August 1st, 2001. After that date all commits to the `RELENG_4` branch of FreeBSD had to be explicitly approved by the `Release`

Engineering Team <re@FreeBSD.org>. The first release candidate for the x86 architecture was released on August 16, followed by 4 more release candidates leading up to the final release on September 18th. The security officer was very involved in the last week of the process as several security issues were found in the earlier release candidates. A total of over *500* emails were sent to the Release Engineering Team <re@FreeBSD.org> in little over a month.

Our user community has made it very clear that the security and stability of a FreeBSD release should not be sacrificed for any self-imposed deadlines or target release dates. The FreeBSD Project has grown tremendously over its lifetime and the need for standardized release engineering procedures has never been more apparent. This will become even more important as FreeBSD is ported to new platforms.

7. Future Directions

It is imperative for our release engineering activities to scale with our growing userbase. Along these lines we are working very hard to document the procedures involved in producing FreeBSD releases.

- *Parallelism* - Certain portions of the release build are actually "embarrassingly parallel". Most of the tasks are very I/O intensive, so having multiple high-speed disk drives is actually more important than using multiple processors in speeding up the `make release` process. If multiple disks are used for different hierarchies in the chroot(2) environment, then the CVS checkout of the ports and doc trees can be happening simultaneously as the `make world` on another disk. Using a RAID solution (hardware or software) can significantly decrease the overall build time.
- *Cross-building releases* - Building IA-64 or Alpha release on x86 hardware? `make TARGET=ia64 release`.
- *Regression Testing* - We need better automated correctness testing for FreeBSD.
- *Installation Tools* - Our installation program has long since outlived its intended life span. Several projects are under development to provide a more advanced installation mechanism. The libh project was one such project that aimed to provide an intelligent new package framework and GUI installation program.

8. Acknowledgements

I would like to thank Jordan Hubbard for giving me the opportunity to take on some of the release engineering responsibilities for FreeBSD 4.4 and also for all of his work throughout the years making FreeBSD what it is today. Of course the release would not have been possible without all of the release-related work done by Satoshi Asami <asami@FreeBSD.org>, Steve Price <steve@FreeBSD.org>, Bruce A. Mah <bmah@FreeBSD.org>, Nik Clayton <nik@FreeBSD.org>, David O'Brien <obrien@FreeBSD.org>, Kris Kennaway <kris@FreeBSD.org>, John Baldwin <jhb@FreeBSD.org> and the rest of the FreeBSD development community. I would also like to thank Rodney W. Grimes <rgrimes@FreeBSD.org>, Poul-Henning Kamp <phk@FreeBSD.org>, and others who worked on the release engineering tools in the very early days of FreeBSD. This article was influenced by release engineering documents from the CSRG [7] , the NetBSD Project, [8] , and John Baldwin's proposed release engineering process notes. [9]

[1] Subversion, http://subversion.apache.org

[2] FreeBSD committers

[3] FreeBSD Core Team

[4] Rebuilding world

[5] FreeBSD Ports Collection https://ports.FreeBSD.org

[6] Diskless Operation with PXE

[7] Marshall Kirk McKusick, Michael J. Karels, and Keith Bostic: The Release Engineering of 4.3BSD

[8] NetBSD Developer Documentation: Release Engineering http://www.NetBSD.org/developers/releng/index.html

[9] John Baldwin's FreeBSD Release Engineering Proposal https://people.FreeBSD.org/~jhb/docs/releng.txt