FreeBSD Release Engineering for Third Party Software Packages

Steve Price <steve@FreeBSD.org >
Revision: 43184

FreeBSD is a registered trademark of the FreeBSD Foundation.

Intel, Celeron, Centrino, Core, EtherExpress, i386, i486, Itanium, Pentium, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

XFree86 is a trademark of The XFree86 Project, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this document, and the FreeBSD Project was aware of the trademark claim, the designations have been followed by the "™" or the "®" symbol.

2013-11-13 by hrs.

Abstract

This paper describes the approach used by the FreeBSD ports management team to produce a high quality package set suitable for official FreeBSD release media. This document is a work in progress, but eventually it will cover the process used to build a clean package set on the FreeBSD.org "Ports Cluster", how to configure any other set of machines as a ports cluster, how to split up the packages for the release media, and how to verify that a package set is consistent.

Table of Contents

1. Building packages from the Ports Collection 2
2. The Package Split 2

1. Building packages from the Ports Collection

The FreeBSD Ports collection is a collection of over 24,000 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 a given FreeBSD release.

1.1. The Ports Cluster

In order to provide a consistent set of third-party packages for FreeBSD releases, every port is built in a separate chroot environment, starting with an empty `/usr/local` and `/usr/X11R6` . The requisite dependencies are installed as packages before the build proceeds. This enforces *consistency* in the package build process. By starting the package build in a pristine environment, we can assure that the package metadata (such as required dependencies) is accurate. This way, we will never generate packages that might work on some systems and not on others depending on what software was previously installed.

The “Ports Cluster” for the x86 architecture currently consists of a master node (Dual Pentium® III 733MHz) and 8 slave nodes (Pentium® III 800MHz) to do the actual package builds. With this configuration, a complete package build takes over 24 hours. These machines are co-located with the other FreeBSD Project equipment at Yahoo's corner of Exodus in Santa Clara, CA.

The “Ports Cluster” for the Alpha architecture consists of 7 PWS 500A machines donated by Compaq and also co-located with Yahoo's facilities.

2. The Package Split

For FreeBSD 4.4 over 4.1 gigabytes of packages were created. This causes a problem for CDROM distributions because we would like to ship as many packages as possible without making the user insert another disc to satisfy dependencies. The solution is to create “clusters” of like packages with similar dependencies and group these onto specific discs. This section describes the software and methodology used to create those package sets for the official FreeBSD release discs.

The scripts and other files needed to produce a package split can be found in the CVS tree in `ports/Tools/scripts/release` . Copy this directory to a machine that has enough free disk space to hold 2 to 3 times the size of the package set that you wish to split.

The following scripts are present in this directory:

`config`
: This file contains the free space on each disc and whether packages, distfiles, or both are allowed on any given disc. The first column is the disc name. It must be of the form `disc[0-9a-z]` . Currently it is set up to allow for 10 discs (4 for the release set

and 6 for the toolkit). There is an implied extra disc called "scratch" where all of the remaining distfiles/packages land if they do not fit elsewhere. The second column can be either a 1 or 0, where 1 says that it is okay to place packages on this disc. The third column works the same way, but it controls whether distfiles are placed on this disc. The last column denotes the number of bytes of free space on a disc.

`doit.sh`
: This is the workhorse. Once you have all the files in place and things properly configured this script directs the process of splitting packages. Beware it is interactive so you need to keep an eye on it as it runs. More details on what happens in this script will follow.

`checkdeps.pl`
: Makes sure all packages dependencies are satisfied given an `INDEX` file and a directory of packages.

`oneshot.pl`
: This is where all the magic (and I use that term loosely as it is mostly just a brute force approach) happens. Given a list of required packages for each disc and a set of packages/distfiles this is the script that places a package or distfile on a disc along with all of its dependencies.

`print-cdrom-packages.sh`
: This file is a copy of `src/release/scripts/print-cdrom-packages.sh` from the release you are working on.

`scrubindex.pl`
: This script removes lines from an `INDEX` file for packages that are not present. It also removes the XFree86™ dependencies. NOTE: you will need to tweak the value of the `xdep` variable to make sure the version number is correct.

`setup.sh`
: This is a helper script that I use on the ports building cluster to grab a copy of the ports tree and the matching set of the packages/distfiles.

Here is a checklist of things you will need to check or configure before going any further.

1. Edit `config` to denote the number of discs you have, their sizes, and whether you want them want to contain packages, distfiles, both, or neither.

2. Make sure you remove the `gen` directory if there is an old one laying around. This directory contains working files that will only be valid for the current split.

3. On your first pass through a split it is best to fake the copying of packages and distfiles. This will save both time and disk space while you do a couple of trial runs to make sure things fit, etc. In the `oneshot.pl` set the `fake` variable to 1 and instead of actually copying the files it will touch(1) them. Be sure you turn this off or set `fake` to 0 before

you give the resultant discs to the person that will be mastering the discs otherwise they will get a directory full of zero-sized files.

4. Make sure you have a recent copy of the `print-cdrom-packages.sh` and that it is from the correct release.

5. Check to make sure the XFree86™ dependency in `scrubindex.pl` has the correct version number. You will also need to make sure this value is correct in `doit.sh` as well.

Next you will need to get a copy of the ports tree, packages, and distfiles from a recent build on the package cluster. See the `setup.sh` for a working example but essentially here is what needs to be done.

1. Grab a copy of `ports.tar.gz` and extract it into the `ports` directory alongside `doit.sh` and the `scripts` directory.

2. Remove the packages/distfiles directories or symlinks. Bento has these as symlinks and you will have mixed results if you do not get rid of them before proceeding.

3. Create a new ports/packages directory and copy the package set from the package building cluster.

4. Create a new ports/distfiles directory and copy the distfiles from the package building cluster. NOTE: if you do not want any distfiles simply create the directory and leave it empty. This directory must be present even if it does not contain anything.

Now we are finally ready for the fun task of actually splitting the packages. You start the processing by running `./doit.sh` . Here is what it does the first time you run it.

1. Create a list of the restricted (can not be on the master FTP site) ports.

2. Asks you if you would like to remove the restricted ports. Most of the time you will want to answer (y)es here.

3. Create a list of the packages/distfiles that can not be put on the discs.

4. Asks you if you would like to remove the non-cdromable packages/distfiles. Most of the time you will want to answer (y)es here.

5. Copies the `INDEX` from the `ports` directory to the `gen` directory. In doing so it removes the lines for ports where the packages do not exist. It also checks to make sure that all of the required dependency packages are present.

6. Create a list of packages that are required on each disc.

7. Asks you if you would like to populate the discs. After populating each disc it will check for missing dependencies, scrub the `INDEX` file, and create the `CHECKSUM.MD5` file.

8. Check to make sure the required packages made it on each disc and gives you a summary of the sizes of each disc.

After going through this the first time if you are lucky enough that all of the required packages built and fit on each disc. All you need to do is set `fake` to `0` in `oneshot.pl` and re-run `./doit.sh`. The second and subsequent times around it will skip steps 1-5 above. If you want to re-run any of those steps refer to `doit.sh` for which files need to be removed to not short-circuit those steps. If you want to repeat all of these steps then the easiest way is to `rm -rf gen`.

Upon successful completion the packages/distfiles will be in the `disc*` directories and the leftover will be in the `scratch` directory.

What to do if things go wrong? Here is some common gotchas and workarounds.

Missing required packages
: This is a pretty common occurrence. You will either need to wait for a new set of packages where the missing packages were built or get someone to re-start the package build for you. *Do not* attempt to build the missing packages on your own machine and add them into the fray. While you might be able to get away with this if you are extremely careful the vast majority of the time you will miss some little detail and the simple process of adding a package could make hundreds of others come up mysteriously broken.

Required packages will not fit
: This happens on occasion too and is relatively easy to fix. Simply edit `print-cdrom-packages.sh` to move packages around until they fit. Yes this is an iterative process and one of the reasons why you should enable `fake` in `oneshot.pl` until you have gotten things the way you want them. Re-run `./doit.sh` after you made your adjustments.

Required packages not on the right (or any) disc
: This usually means you did not add them to `print-cdrom-packages.sh` or you put them on the wrong disc. This script is the gospel by which this whole process determines where a package must be. If you want to force a package to land on a particular disc this is the only way to ensure that it will happen.

If you get completely stuck and can not figure out why things are borked or how to fix them then email Steve Price <steve@FreeBSD.org> for assistance.