bsd_day(2011)

FreeBSD Ports and Packages – Getting Back Being the Best

- A short journey into the FreeBSD Ports Collection: past, present and future -

By

Ion-Mihai “IOnut” Tetcu

FreeBSD committer, itetcu@FreeBSD.org

Ports and packages

FreeBSD is bundled with a rich collection of system tools as part of the base system. However, there is only so much one can do before needing to install an additional third-party application to get real work done.

FreeBSD provides two complementary technologies for installing third-party software on your system: the FreeBSD **Ports Collection** (for installing from source), and **packages** (for installing from pre-built binaries).

Either method may be used to install the newest version of your favorite applications from local media or straight off the network.

FreeBSD Ports and Packages

A **package** is a *binary file* that includes *the files of an application* (binaries, shared libraries, manual pages, examples, data files, etc.) and *installation/deinstallation commands*.

A **port** is a *collection of files* that allow an application to be *compiled* from source (if applicable) and *installed and/or deinstalled* with minimal hustle for the user.

The Ports Collection (sometimes referenced to as The Ports Tree, PT or simply "the ports") is comprised of the *ports themselves* organized into 63 categories and a *framework* (Mk/bsd.*.mk) with make(1) instructions which are common to the process of building more that a few ports.

Both ports and packages understand the concept of **dependencies** – other third-party applications that are needed for a specific application in order to function.

It all started with this:

CVS log for ports/Makefile

Revision 1.1.1.1 (vendor branch):

Sun Aug 21 13:19:26 1994 UTC (17 years, 2 months ago) by jkh

Branches: ports

CVS tags: ports_2_0

The start of the 2.0 ports collection. No sup repository yet, but I'll make one when I wake up again.. :)

Revision 1.1:

Sun Aug 21 13:19:25 1994 UTC (17 years, 2 months ago) by jkh

Branches: MAIN

Initial revision

Which contained this:

```
SUBDIR=........editors shells

.include <bsd.subdir.mk>
```

And now:

Revision 1.109:

Wed May 4 22:33:13 2011 UTC (6 months ago) by flz

Branches: MAIN

CVS tags: HEAD

Changes since revision 1.108: +6 -2 lines

Latest round of infrastructure changes:

[..]

```
# wc -l /usr/ports/Makefile
    192 /usr/ports/Makefile
```

Stats:

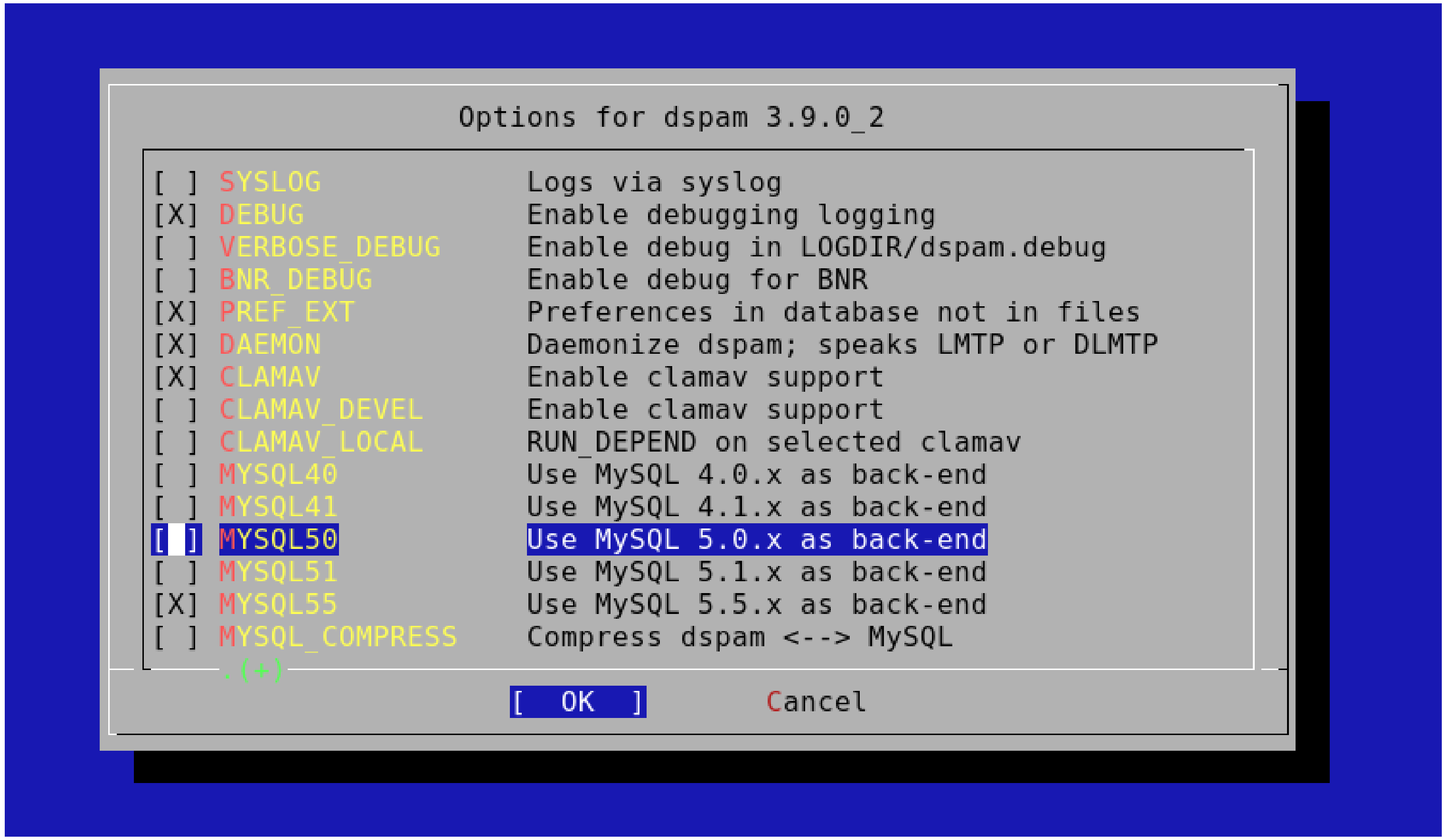
- ~ 23 000 ports
- about 55GB of current DISTFILES for the whole PT
- 22182 LOC in ports/Mk/bsd.*.mk files
- in 49 files
- ~ 70GB total packages size for one release

Ports are nice because:

- They allow you to fine-tune what you want from a given app or set of apps

... but they take a lot of time to compile

OPTIONS provide a friendly way to customize ports:



```
                    Options for dspam 3.9.0_2

 [ ] SYSLOG            Logs via syslog
 [X] DEBUG             Enable debugging logging
 [ ] VERBOSE_DEBUG     Enable debug in LOGDIR/dspam.debug
 [ ] BNR_DEBUG         Enable debug for BNR
 [X] PREF_EXT          Preferences in database not in files
 [X] DAEMON            Daemonize dspam; speaks LMTP or DLMTP
 [X] CLAMAV            Enable clamav support
 [ ] CLAMAV_DEVEL      Enable clamav support
 [ ] CLAMAV_LOCAL      RUN_DEPEND on selected clamav
 [ ] MYSQL40           Use MySQL 4.0.x as back-end
 [ ] MYSQL41           Use MySQL 4.1.x as back-end
 [ ] MYSQL50           Use MySQL 5.0.x as back-end
 [ ] MYSQL51           Use MySQL 5.1.x as back-end
 [X] MYSQL55           Use MySQL 5.5.x as back-end
 [ ] MYSQL_COMPRESS    Compress dspam <--> MySQL
       .(+)
                  [  OK  ]        Cancel
```

Friendly?

OMG!

That looks sooo 80sh!

But!

It's much nicer that having to read the Makefile ;)

```
$ wc -l /usr/ports/mail/dspam/Makefile
     639 /usr/ports/mail/dspam/Makefile
```

But even worse:

- For each OPTION we have only on / off (not much option there, actually)
- No support for exclusive or dependent options
- Inconsistent naming in different ports for the same thing
- pretty much impossible to enforce globaly

OPTIONS-NG – types:

- simple options (on / off)
- single options (1 and only one in a group of options)
- multi options (at least 1 from a group of options)
- system wide options
- per port options
- generic, general options descriptions

OPTIONS-NG - consistency

- automatic consistency check: check-config target
- default global options
- OPTIONS_EXCLUDE: to exclude global options that the port doesn't support
- OPTIONS_DEFAULT: for the port's defaults
- OPTIONS_[SET|UNSET]: for the user to set globaly

- backward-compatible

OPTIONS-NG – Makefile example

```
OPTIONS_DEFINE= OPT1 OPT2 OPT3

OPTIONS_MULTI= GRP1 GRP2
OPTIONS_MULTI_GRP1= OPT4 OPT5
OPTIONS_MULTI_GRP2= OPT6 OPT7

OPTIONS_SINGLE= SEL1
OPTIONS_SINGLE_SEL1= OPT8 OTP9 OPT10

OPTIONS_DEFAULT= OPT2 OPT3 OPT9 OPT7 OPT8 OPT4
OPTIONS_EXCLUDE= NLS DOCS
OPT1_DESC= "Description of my option"
```

OPTIONS-NG – make showconfig

```
===> The following configuration options are available:
OPT1=off: Description of my option
OPT2=on
OPT3=on
====> Options available for the multi GRP1: you must choose at least one of them
OPT4=on
OPT5=off
```

OPTIONS-NG – make showconfig (continued)

```
====> Options available for the multi GRP2: you
must choose at least one of them
OPT6=off
OPT7=on
====> Options available for the single SEL1: you
must select only one of them
OPT8=on
OPT9=off
OPT10=off
```

OPTIONS-NG – make pretty-print-config

```
# make pretty-print-config
 +DOCS +NLS -A -B -C -D -E \
 A[ -M -N -O ] G[-P -Q ] \
 B( -R -Z ) F( -Y -W )
```

OPTIONS-NG

...............Questions?

packages are nice because:

- they are fast to fetch and install

pacakges – the tools to work with them aren't

```
src/usr.sbin/pkg_install/add/perform.c
/*
 * This is seriously ugly code following.  Written very
 * fast![And subsequently made even worse..  Sigh!
 * This code was just born to be hacked, I guess.. :) ]
 */

 --
 jhk@
 Jordan K. Hubbard
 18 July 1993
```

pkg* – problems:

- code hard to maintain / extend
- no safe upgrades
- lack features provides by ports
- no real repository handling
- checks done very late (at install time)
- slow when many packages are installed (flat text files database, for 924 packages: 643 files with 1 400 626 lines)

pkgng is:

- a binary package management (like apt/yum/pacman)
- a tool to query/manage installed packages
- a tool to deal with binary packages
- a tool to safetly upgrade/install packages from a remote repository
- a library that provides all the package management in a safe way so one can write easily and safetly a new frontend
- compatible with exisitng ports

pkgng is:

http://www.youtube.com/watch?v=IRa6wFBLU28

http://people.freebsd.org/~bapt/pkgng-upgrade.avi

pkgng – libpkg

- does **all** the work: everything to deal with packages and to create packages, to deal and to create repositories
- simple and consistent API, easy-to-write binding
- thread-safe
- clean and simple code base, easy to maintain and extand

pkgng – local packages database

- installed packages registered in a SQLite database → fast queries and ACID transactions.
- records if packages were installed as dependencies of other packages (whether from local ports or from the remote repositories.) autoremove proposes to remove those automatic packages if no others depend on them. It also prints the disk space saved.
- backup: exports/restores the whole database

pkgng – package format

- More data: OPTIONS, package size, both compressed and flat, license
- YAML manifest
- upgrade scripts

pkgng – CLI: pkg

“Local” subcommands:

- create: create a pacakge in a chroot
- add: install/upgrade a package
- register: register a package installed from ports
- info: shows information about an installed package
- which: tells what package installed a given file
- delete: removes a package

pkgng – CLI: pkg

“Remote” subcommands:

- fetch: fetch an updated version of the repository
- upgrade: finds all the installed packages that can be updated and installs the new dependencies if needed.
- search: show info about a package (from a remote repository)
- install: installs sets of packages from a remote repository

pkgng – remote (repository) database

- Everything but files stored in SQLite DB
- RSA signed
- .tar.xz
- pkg repo my_dir [my_rsa_key]

pkgng

http://wiki.freebsd.org/pkgng
https://github.com/pkgng/pkgng

Authors:
Baptiste Daroussin <bapt@FreeBSD.org>
Julien Laffaye <jlaffaye@FreeBSD.org>

Starting point: NetBSD's pkgin

pkgng

Questions?

QA processes for the Ports Collections

The sope of the problem:

- we have about 23,000 ports (and the number keeps increasing; we added more than 500 in the last 6 months).
- we have 2 tier-one architectures (soon with two more): i386, amd64
- we have 3 supported OS versions: 8.x, 9.x and CURRENT

QA

The simple matrix based on these 3 considerations starts to look a little overwhelming.

But when we start to add the global options, per port specific options, and different possible application versions we quickly go from a 3-dimensional matrix to a n-dimensional one, with a rather large value of “n”.

QA - PortsMon

Database that processes information from several sources and allows its to be browsed via a web interface. Currently, the ports Problem Reports (PRs), the error logs from the build cluster, and individual files from the ports collection are used. In the future, this will be expanded to include the distfile survey, as well as other sources.

QA – vdiff

Most people are born with a Mk1 Eyeball, in fact they are normally born with a pair of Mk1 Eyeballs. These are excellent observing tools, sensitive across the whole of the visible spectrum (now there's a coincidence!) and equipped with a complex, parallel processing, image analysis and recognition system (know as "The Brain"). Experts in pattern recognition maintain that Mk1 Eyeballs used together with The Brain are the most effective tools for data analysis.

FreeBSD Ports and Packages

Committing with them disconnected from The Brain is an excellent way of acquiring Pointyhats, after which people in the committer's vicinity usually hear a big noise (produced by a hard slap of The Brain Case - used probably in order to reconnect the two devices) and the muttering “How the h[...] didn't I see that?!”

Pointy Hat: the mythical dunce cap awarded to FreeBSD committers who commit something to the source tree that doesn't work, especially if it doesn't build in the first place.

QA during the release process

Around BETA or RC stage, the Ports Tree is frozen. This means than all commits have to be approved by the Ports Management Team (except those done by the Security Team which has a blanket approval from portmgr).

During this freeze period the maintainers' and committers' energy is spent on fixing bugs rather that updating ports to new versions.

Numerous test builds are done on all supported architectures on the upcoming OS version to insure that problems are identified and fixed.

Daily QA

- Pointyhat
- QAT
- user feedback

Pointyhat

- two clusters (-east and -west).
- used for pacakge building
- used for experimental runs (changes in infrastructure / high impact ports).

QAT (QA Tinderbox or QA Tindy)

Commit-triggered builds:

- the QATMail sent to the committer, the port's maintainer and the original recipients of the cvs commit mail (cvs* mailing lists);
- each commit-triggered build that fails warrants a QATMail

QAT (QA Tinderbox or QA Tindy)

For regular and dependencies builds:

- the email is sent to the maintainer of the port
- with the Subject: of the email of the form “category/port - fails: fail_reason"
- a limit of one email / port / week is set in order to avoid bothering maintainers too much

QA – User feedback

At times incompleate, nagging, annoying but

PRICELESS

QA

Questions?