NAME

ports - contributed applications

DESCRIPTION

The FreeBSD Ports Collection offers a simple way to compile and install third party applications. It is also used to build packages, to be installed using pkg(8).

The ports tree, typically located at */usr/ports*, consists of subdirectories, one for each category; those in turn contain individual ports. Each port is a directory with metadata and patches necessary to make the original application source code compile and run on FreeBSD. Compiling an application is as simple as typing "make build" in the port directory. The *Makefile* automatically fetches the application source code, either from a local disk or the network, unpacks it, applies the patches, and compiles it. It also recursively handles dependencies -- other pieces of software the port depends on in order to build and work. Afterwards, "make install" installs the application.

The FreeBSD Ports Collection is maintained in several branches, which differ mostly by versions of software provided: the *main* branch contains all the latest changes and corresponds to the *latest* package set, while the *quarterly* branches only provide critical fixes. The *main* branch can be cloned and updated from the Git repository located at:

https://git.FreeBSD.org/ports.git

so eg:

git clone https://git.FreeBSD.org/ports.git

The *quarterly* branches can be found in Git as branches like *yyyyQn* , where *yyyy* indicates the year and *n* indicates the quarter (1 to 4), eg:

git clone -b 2021Q2 https://git.FreeBSD.org/ports.git

It is generally a good idea to use the **ports** branch that matches the pkg(8) repository being used. By default, for FreeBSD CURRENT the pkg(8) is configured to install packages built from the *main* branch, while for FreeBSD STABLE or RELEASE versions it is configured to install packages built from the latest *quarterly* branch. Currently configured pkg(8) repository can be verified by looking at the *url* field in **pkg -vv** output.

For more information about using ports, see the "Packages and Ports section" in *The FreeBSD Handbook*:

https://docs.FreeBSD.org/en/books/handbook/ports/

For information about creating new ports, see *The Porter's Handbook*:

https://docs.FreeBSD.org/en/books/porters-handbook/

TARGETS

Some of the make(1) targets work recursively through subdirectories. This lets you, for example, install all of the "biology" ports with one command. The targets that do this are **build**, **checksum**, **clean**, **configure**, **depends**, **extract**, **fetch**, **install**, and **package**.

The following targets will be run automatically by each proceeding target in order. That is, **build** will be run (if necessary) by **install**, and so on all the way to **fetch**. Usually, you will only use the **install** target.

config Configure *OPTIONS* for this port using dialog4ports(1).

fetch Fetch all of the files needed to build this port from the sites listed in *MASTER_SITES* and *PATCH_SITES*. See *FETCH_CMD*, *MASTER_SITE_OVERRIDE* and *MASTER_SITE_BACKUP*.

checksum
Verify that the fetched distfile's checksum matches the one the port was tested against. If the distfile's checksum does not match, it also fetches the distfiles which are missing or failed the checksum calculation. Defining *NO_CHECKSUM* will skip this step.

depends Install (or compile if only compilation is necessary) any dependencies of the current port. When called by the **extract** or **fetch** targets, this is run in piecemeal as **fetch-depends**, **build-depends**, etc. Defining *NO_DEPENDS* will skip this step.

extract Expand the distfile into a work directory.

patch Apply any patches that are necessary for the port.

configure Configure the port. Some ports will ask you questions during this stage. See *INTERACTIVE* and *BATCH*.

build Build the port. This is the same as calling the **all** target.

install Install the port and register it with the package system. This is all you really need to do.

install-missing-packages
Install missing dependencies from packages instead of building them.

The following targets are not run during the normal install process.

showconfig Display *OPTIONS* config for this port.

showconfig-recursive
Display *OPTIONS* config for this port and all its dependencies.

rmconfig Remove *OPTIONS* config for this port.

rmconfig-recursive
Remove *OPTIONS* config for this port and all its dependencies.

config-conditional
Skip the ports which have already had their *OPTIONS* configured.

config-recursive
Configure *OPTIONS* for this port and all its dependencies using dialog4ports(1).

fetch-list Show the list of files to fetch in order to build the port (but not its dependencies).

fetch-recursive Fetch the distfiles of the port and all its dependencies.

fetch-recursive-list
Show list of files that would be retrieved by **fetch-recursive**.

build-depends-list, **run-depends-list**
Print a list of all the direct compile or run dependencies for this port.

all-depends-list
Print a list of all recursive dependencies for this port.

pretty-print-build-depends-list, **pretty-print-run-depends-list**
Print a list of all the recursive compile or run dependencies for this port by port name and version.

missing Print a list of missing dependencies to be installed for the port.

clean Remove the expanded source code. This recurses to dependencies unless *NOCLEANDEPENDS* is defined.

distclean Remove the port's distfiles and perform the **clean** target. The **clean** portion recurses to dependencies unless *NOCLEANDEPENDS* is defined, but the **distclean** portion never recurses (this is perhaps a bug).

reinstall Use this to restore a port after using pkg-delete(8) when you should have used **deinstall**.

deinstall Remove an installed port from the system, similar to pkg-delete(8).

deinstall-all Remove all installed ports with the same *PKGORIGIN* from the system.

package Make a binary package for the port. The port will be installed if it has not already been. The package is a *.pkg* file that you can use to install the port on other machines with pkg-add(8). If the directory specified by *PACKAGES* does not exist, the package will be put in */usr/ports/category/port/work/pkg*. See *PKGREPOSITORY* and *PKGFILE* for more information.

package-recursive
Like **package**, but makes a package for each depending port as well.

package-name Prints the name with version of the port.

readmes Create a port's *README.html*. This can be used from */usr/ports* to create a browsable web of all ports on your system!

search Search the *INDEX* file for the pattern specified by the *key* (searches the port name, comment, and dependencies), *name* (searches the port name only), *path* (searches the port path), *info* (searches the port info), *maint* (searches the port maintainer), *cat* (searches the port category), *bdeps* (searches the port build-time dependency), *rdeps* (searches the port run-time dependency), *www* (searches the port web site) make(1) variables, and their exclusion counterparts: *xname*, *xkey* etc. For example, one would type:

```
cd /usr/ports && make search name=query
```

to find all ports whose name matches "query". Results include the matching ports' path, comment, maintainer, build dependencies, and run dependencies.

```
cd /usr/ports && make search name=pear- \
  xbdeps=apache
```

To find all ports whose names contain "pear-" and which do not have apache listed in build-time dependencies.

```
cd /usr/ports && make search name=pear- \
  xname='ht(tp|ml)'
```

To find all ports whose names contain "pear-", but not "html" or "http".

```
make search key=apache display=name,path,info keylim=1
```

To find ports that contain "apache" in either of the name, path, info fields, ignore the rest of the record.

By default the search is not case-sensitive. In order to make it case-sensitive you can use the *icase* variable:

```
make search name=p5-R icase=0
```

quicksearch Reduced **search** output. Only display name, path and info.

describe Generate a one-line description of each port for use in the *INDEX* file.

maintainer Display the port maintainer's email address.

index Create */usr/ports/INDEX*, which is used by the **pretty-print-*** and **search** targets. Running the **index** target will ensure your *INDEX* file is up to date with your ports tree.

fetchindex Fetch the *INDEX* file from the FreeBSD cluster.

ENVIRONMENT

You can change all of these.

PORTSDIR Location of the ports tree. This is */usr/ports* by default.

WRKDIRPREFIX

Where to create any temporary files. Useful if *PORTSDIR* is read-only (perhaps mounted from a CD-ROM).

DISTDIR Where to find/put distfiles, normally *distfiles/* in *PORTSDIR*.

SU_CMD Command used to elevate privilege to configure and install a port. The unprivileged user must have write access to *WRKDIRPREFIX* and *DISTDIR*. The default is '/usr/bin/su root -c'. Many users set it to '/usr/local/bin/sudo -E sh -c' for convenience.

PACKAGES Used only for the **package** target; the base directory for the packages tree, normally *packages/* in *PORTSDIR*. If this directory exists, the package tree will be (partially) constructed. This directory does not have to exist; if it does not, packages will be placed into the current directory, or you can define one of

PKGREPOSITORY Directory to put the package in.

PKGFILE The full path to the package.

LOCALBASE Where existing things are installed and where to search for files when resolving dependencies (usually */usr/local*).

PREFIX Where to install this port (usually set to the same as *LOCALBASE*).

MASTER_SITES Primary sites for distribution files if not found locally.

PATCH_SITES Primary locations for distribution patch files if not found locally.

MASTER_SITE_FREEBSD
If set, go to the master FreeBSD site for all files.

MASTER_SITE_OVERRIDE
Try going to these sites for all files and patches, first.

MASTER_SITE_BACKUP
Try going to these sites for all files and patches, last.

RANDOMIZE_MASTER_SITES
Try the download locations in a random order.

MASTER_SORT Sort the download locations according to user supplied pattern. Example:
.dk .sunet.se .se dk.php.net .no .de heanet.dl.sourceforge.net

MASTER_SITE_INDEX
Where to get *INDEX* source built on FreeBSD cluster (for **fetchindex** target). Defaults to *https://www.FreeBSD.org/ports/*.

FETCHINDEX Command to get *INDEX* (for **fetchindex** target). Defaults to "fetch -am".

NOCLEANDEPENDS
If defined, do not let **clean** recurse to dependencies.

FETCH_CMD Command to use to fetch files. Normally fetch(1).

FORCE_PKG_REGISTER
If set, overwrite any existing package registration on the system.

INTERACTIVE If defined, only operate on a port if it requires interaction.

BATCH If defined, only operate on a port if it can be installed 100% automatically.

DISABLE_VULNERABILITIES
If defined, disable check for security vulnerabilities using pkg-audit(8) when installing new ports.

NO_IGNORE If defined, allow installation of ports marked as <*FORBIDDEN*>. The default behavior of the Ports framework is to abort when the installation of a forbidden port is attempted. Of course, these ports may not work as expected, but if you really know what you are doing and are sure about installing a forbidden port, then *NO_IGNORE* lets you do it.

NO_CHECKSUM
If defined, skip verifying the port's checksum.

TRYBROKEN If defined, attempt to build a port even if it is marked as <*BROKEN*>.

PORT_DBDIR Directory where the results of configuring *OPTIONS* are stored. Defaults to */var/db/ports*. Each port where *OPTIONS* have been configured will have a uniquely named sub-directory, containing a single file *options*.

MAKE VARIABLES

The following list provides a name and short description for many of the variables that are used when building ports. More information on these and other related variables may be found in

*${PORTSDIR}/Mk/** and the FreeBSD Porter's Handbook.

WITH_DEBUG (*bool*) If set, debugging symbols are installed for ports binaries.

WITH_DEBUG_PORTS A list of origins for which to set *WITH_DEBUG*.

DEBUG_FLAGS (Default: '-g') Additional *CFLAGS* to set when *WITH_DEBUG* is set.

WITH_CCACHE_BUILD (*bool*) If set, enables the use of ccache(1) for building ports.

CCACHE_DIR Which directory to use for the ccache(1) data.

FILES

/usr/ports The default ports directory.
/usr/ports/Mk/bsd.port.mk The big Kahuna.

EXAMPLES

Example 1: Building and Installing a Port

The following command builds and installs Emacs.

```
# cd /usr/ports/editors/emacs
# make install
```

Example 2: Installing Dependencies with pkg(8)

The following example shows how to build and install a port without having to build its dependencies. Instead, the dependencies are downloaded via pkg(8).

```
# make install-missing-packages
# make install
```

It is especially useful, when the dependencies are costly in time and resources to build (like *lang/rust*). The drawback is that pkg(8) offers only packages built with the default set of *OPTIONS*.

Example 3: Building a Non-Default Flavor of a Port

The following command builds a non-default flavor of a port. (In this case *devel/py-pip* is going to be built with Python 3.7 support.)

```
# cd /usr/ports/devel/py-pip
# env FLAVOR=py37 make build
```

Example 4: Setting Ports Options via make.conf(5)

The following lines present various ways of configuring ports options via make.conf(5) (as an alternative to, e.g., running "make config"):

```
# Enable NLS for all ports unless configured otherwise
# using the options dialog.
OPTIONS_SET=            NLS
# Disable DOCS for all ports overriding the options set
# via the options dialog.
OPTIONS_UNSET_FORCE=        DOCS
# Disable DOCS and EXAMPLES for the shells/zsh port.
shells_zsh_UNSET=       DOCS EXAMPLES
```

These and other options-related variables are documented in */usr/ports/Mk/bsd.options.mk*.

Example 5: Setting make(1) Variables for Specific Ports via make.conf(5)

The following example shows how to set arbitrary make(1) variables only specific ports:

```
# Set DISABLE_MAKE_JOBS for the lang/rust port:
.if ${.CURDIR:M*/lang/rust}
DISABLE_MAKE_JOBS=  yes
TRYBROKEN=          yes
.endif
```

Example 6: Debugging Ports

By default ports are built and packaged without debugging support (e.g., debugging symbols are stripped from binaries, optimization flags are used for compiling, verbose logging is disabled). Whether ports are built with debugging symbols can be controlled by the settings in make.conf(5), e.g.,

```
# Enable debugging for all ports.
WITH_DEBUG=         yes
# Enable debugging for selected ports.
WITH_DEBUG_PORTS=   mail/dovecot security/krb5
```

It is also possible to use the debug variables on the command line:

make -DWITH_DEBUG DEBUG_FLAGS="-g -O0" build

See the *MAKE VARIABLES* section to learn more about the debug variables.

To understand the details of what happens when the debug variables are set it is best to consult the files located at *${PORTSDIR}/Mk/** (*bsd.port.mk* in particular).

If debugging is enabled for a specific port, the ports framework will:

- Add *DEBUG_FLAGS* (defaults to '-g') to *CFLAGS*.
- Try to prevent the binaries from being stripped (including checking the install target to replace 'install-strip' with 'install'). Whether a binary has been stripped can be checked with file(1).
- Try to enable other debugging features like debug build type or verbose logging. However, this is port-specific and the ports framework might not be aware of each supported debugging feature a given piece of software has to offer).

SEE ALSO

make(1), make.conf(5), development(7), pkg(7)

Additional developer documentation:

-portlint(1)

-*/usr/ports/Mk/bsd.port.mk*

Additional user documentation:

-pkg(8)

-*Searchable index of all ports*: **https://www.FreeBSD.org/ports**

HISTORY

The Ports Collection appeared in FreeBSD 1.0. It has since spread to NetBSD and OpenBSD.

AUTHORS

This manual page was originated by David O'Brien.

BUGS

Ports documentation is split over four places -- */usr/ports/Mk/bsd.port.mk*, *The Porter's Handbook*, the "Packages and Ports" chapter of *The FreeBSD Handbook*, and this manual page.