A pkgsrc Bootstrapping Wizard

Contents

The pkgsrc bootstrapping system is a remarkable piece of software which supports virtually any POSIX environment and most popular compiler suites.

However, the inherent complexity of supporting environments ranging from microcontrollers to HPC clusters precludes many of the simplifying assumptions enjoyed by systems supporting only a single platform. Users with such widely varying needs must have a greater technical understanding of their environment and how pkgsrc works internally in order to make optimal choices about their pkgsrc configuration.

For example, what should be used as the base C, C++, and Fortran compilers on which the pkgsrc installation is built? Will there be multiple pkgsrc trees installed on the same system? If so, how does the user select one? Does the system provide X11 or should pkgsrc build its own? Do we want to build portable binary packages, or binaries optimized for the local CPU? What software licenses are acceptable for the organization? Is it OK to install packages with known vulnerabilities?

Using the standard pkgsrc bootstrap script, one would need to learn about all these options and how to configure them in mk.conf before and after the bootstrapping process.

The auto-pkgsrc-setup script embeds the knowledge needed to make some of the most critical decisions while bootstrapping pkgsrc. It detects available compilers and asks the necessary questions to guide the user through the process and generate an installation optimal for their needs.

In doing so, it helps new pkgsrc users get up and running quickly, while also educating them on some of the most important configuration options offered by pkgsrc. Experienced users can also save time and avoid oversights by using auto-pkgsrc-setup instead of bootstrapping manually.

Caution

The bmake binary and other build tools produced by pkgsrc are patched with hard-coded paths under ${PREFIX}. Hence, a given bmake run will always install to the PREFIX for which it was built, rather than the prefix associates with the current working directory. E.g., if you run /usr/pkg-2020Q3/bin/bmake from /usr/pkgsrc-2019Q1/devel/gcc8, the 2019Q1 packages built will be installed under /usr/pkg-2020Q3, a serious problem that will be very difficult to clean up. To mitigate this potential disaster, auto-pkgsrc-setup script generates a wrapper script called "safe-bmake" and a link called "sbmake" for easier typing. This wrapper verifies that your process is in the correct pkgsrc tree for the first bmake binary in your PATH and halts with an error if not.
It is strongly recommended that you use safe-bmake or sbmake for all pkgsrc builds rather than run bmake directly.

Auto-pkgsrc-setup automatically generates a binary bootstrap kit; a tar ball of the pristine installation, so that it can be quickly replicated on other systems running the same operating system.

The primary scripts and modules scrub the environment (PATH, LD_LIBRARY_PATH, CFLAGS, etc.) of any non-system directories to protect against leakage of tools and libraries installed by other package managers or via ad hoc methods. These scripts and modules should always be used for building packages unless you are certain that you want the build to depend on something from outside pkgsrc and the base system.

A second set of identical scripts and modules, suffixed with -non-exclusive, allow other non-standard directories to remain in the path. These can be used when you want access to a pkgsrc installation and other installed software from the same session.

If creating an installation for which there are binary packages avaialable, auto-pkgsrc-setup will automatically install and configure pkgin, with https enabled. Binary packages for NetBSD and RHEL/CentOS for each quarterly snapshot are available here.

Auto-pkgsrc-setup offers the option of using a pkgsrc GCC compiler suite for most pkgsrc builds. This is important on systems such as enterprise Linux, which use older kernels and build tools for reliability, security, and long-term binary compatibility for closed-source applications. For example, RHEL and CentOS 7, the mainstream enterprise Linux systems at the time of this writing, provide GCC 4.8 as the standard compiler via their Yum package manager. In contrast, the current stable GCC release is 9.1.

GCC 4.8 is not capable of building many of the packages provided by pkgsrc. This is due to decisions by upstream developers to use features only available in newer standards such as c++14 or c++17. One can manually build a newer GCC suite and bootstrap and run pkgsrc from this compiler, but this method is difficult to do and very difficult to reproduce faithfully. An ad hoc installation of GCC may produce different results on even slightly different systems as the configure scripts detect the presence of different tools and libraries.

Using a pkgsrc GCC compiler is much easier and more reliable as the pkgsrc GCC packages ensure consistency across many environments. Auto-pkgsrc-setup will carefully craft your mk.conf to use the base GCC only to build a pkgsrc GCC and its dependencies, while all other packages will use the pkgsrc GCC.

Lastly, auto-pkgsrc-setup generates Bourne and C shell scripts as well as environment modules for setting PATH and other important environment variables necessary to use the installation. These include etc/pkgsrc.sh, etc/pkgsrc.csh, and etc/modulefiles/pkgsrc/<snapshot>) for activating the pkgsrc installation, as well as scripts and modules for each GCC package (gcc5.sh, etc.). These are particularly useful when building software outside pkgsrc using pkgsrc tools, such as when using **install.packages()** in a pkgsrc R installation. The scripts and modules relating to the default pkgsrc compiler are linked to gcc-base*, so the user need not know which compiler was selected during bootstrapping. For example:

```
shell-prompt: source /usr/pkg/etc/pkgsrc.sh
shell-prompt: source /usr/pkg/etc/gcc-base.sh   # Ensure that R uses pkgsrc gcc
shell-prompt: R
install.packages("deseq");
```

Note
There are many R-CRAN packages in the pkgsrc collection, as well as a tool for automatically creating them in pkgtools/R2pkg. It may bemore reliable to use pkgsrc to install your R packages rather than running install.packages() under R.

If you're ready to try auto-pkgsrc-setup, just download the script and run it. It can be run as root or an ordinary user and will behave identically in either case, except for the default installation prefix offered. You can choose any prefix that is writable by the user running the script.

You can also use auto-pkgsrc-setup as a knowledge base, documenting the intricacies of installing and configuring pkgsrc. The script is written entirely in POSIX Bourne shell, so the commands for building and configuring pkgsrc are clearly visible.

For bug reports and feature requests, please visit the Github site.