21. Chaos Communications Congress, 27.-29. Dezember 2004, Berlin

Easy Software-Installation on Linux, Solaris, NetBSD etc. using pkgsrc

Hubert Feyrer <hubertf@pkgsrc.org>

December 13, 2004

Abstract

The article discusses the problems when installing open source software on Unix(like) systems and identifies specific areas that need attention, and how they manifest in various architectures of open source systems today, leading from a rather simple layered theory to a complex graph in reality, which requires environmental considerations like demands for flexibility and maintainability when addressed. The pkgsrc system is introduced as a possible solution, which can be used to install software easily from source, independent of your operating system. A general overview of the pkgsrc system is given followed by an user-oriented example on how to bootstrap it and compile packages on a Linux system with a special emphasis of working without root privileges. Operation of the pkgsrc system is described next, with details of the install process and an overview of available packages. The article is intended for users of all Unix(like) systems that need to maintain and update software on a frequently and across various platforms, emphasizing the cross-platform nature of pkgsrc, which includes Linux, FreeBSD, OpenBSD, MacOS X, Solaris, Irix and even MS Windows. .

Contents

1 Introduction

This article contains information abouth the general problems encountered when installing and managing open source software, and introduces the pkgsrc system, which can be used to install software easily from source, independent of your operating system. Instead of knowing details like xmkmf, autoconf, libtool & Makefiles, a simple "make install" is enough to install a package (and all its dependencies). The pkgsrc system will download the package's sources, which is then unpacked, patched, configured, compiled and installed for later querying and removing. The pkgsrc system is based on the NetBSD Packages Collection and was ported to a number of other operating systems like Linux, FreeBSD, OpenBSD, MacOS X, Solaris, Irix and even MS Windows.

2 Issues managing Open Source software

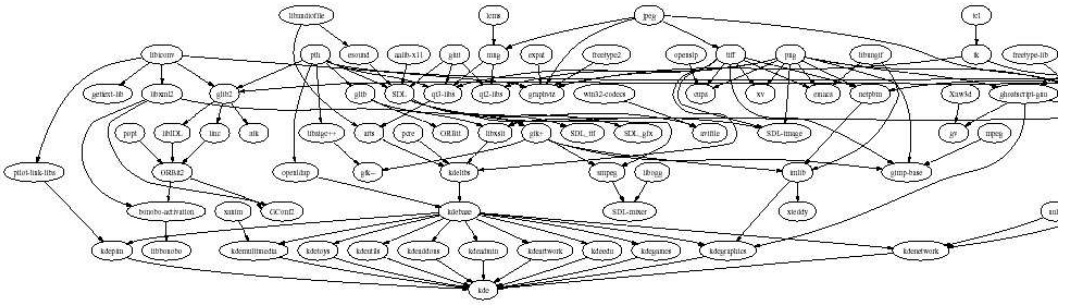
Installation of Open Source software on Unix and Unix-like systems has a number of problems. First and foremost, there are many programs and lots of version changes. Next, compilation costs time – everyone who has tried to compile OpenOffice or KDE knows that these packages still need hours even on latest PC systems. Getting them going on slower, older or non-PC hardware still is an adventure! The fact that software often is not written with portability in mind doesn't aid to this, especially if you're not on a PC running Linux, but we don't want to give a c0ding lesson here.

The installation of software on Unix(like) systems is not trivial either:

- Some basic knowledge about tools is necessary
- There are various ways to configure things (GNU autoconf, Imake, ...)
- There are many side effects depending on other installed packages, compiler switches, etc.

- Many inter-depending packages
- Troubleshooting requires expert knowledge

To illustrate the complexity of inter-depending packages, here is a package dependency graph created from a pkgsrc system running NetBSD[1]:



The bottom of this graph shows KDE as a big package requiring many smaller and small packages, which are placed towards the top of the graph. The obvious complexity of the graph comes from the modularity of Open Source software, where many small packages are used by bigger packages. The complexity is independent of packaging system and operating system, similar graphs can be created for each Linux distribution using its preferred packages system.

The solution to this situation depends on the kind of application. In general, a separation between the base "operating system" and added "applications" need to be made, and depending on the working environment needed, and there are a number of choices. Classical Unix(like) systems are rather small systems that don't come with many applications, but require manual installation of all software. While this is very difficult to install and needs a lot of know-how, this also leads to a very flexible software management that is easy to maintain, even without depending on a vendor providing updates packages. On the other end of the scale are systems that completely integrate applications and operating system, which leads to easy installation, but if manual installation of upgrades are required for parts of the system, they usually evolve into maintenance nightmares. A solution in-between are hybrid systems that come as rather small base operating systems, and which allow adding software packages easily depending on the kind of application, e.g. installing a web server will need other software than a desktop machine or database server. These systems are usually easy to install, and with the aid of a decent packages system, they are easy to maintain. Figure 1 illustrates the degrees of integration between operating systems and applications.

So, where do you want to go today?

- **Easy Installation:** choose this if your software doesn't change often. Use ready-to-user binary distribution. E.g. for desktop systems install Windows or SuSE Linux from CD/DVD.
- **Easy Maintenance:** choose this if you have few packages that change a lot. Take a stable base operating system, and install important packages on your own, e.g. compile on your own on a webserver with Solaris, Apache and PHP.
- **Both:** Welcome to pkgsrc!

There are a number of fine packages systems out there that allow easy installation of applications, and most of these systems are targeted towards one specific operating system or operating system distribution (usually found in Linux land). Few of these systems work on more than one operating system, and pkgsrc is introduced here as a packages system that supports a wide number of platforms.

[1]Made using pkgdepgraph and dot/graphviz

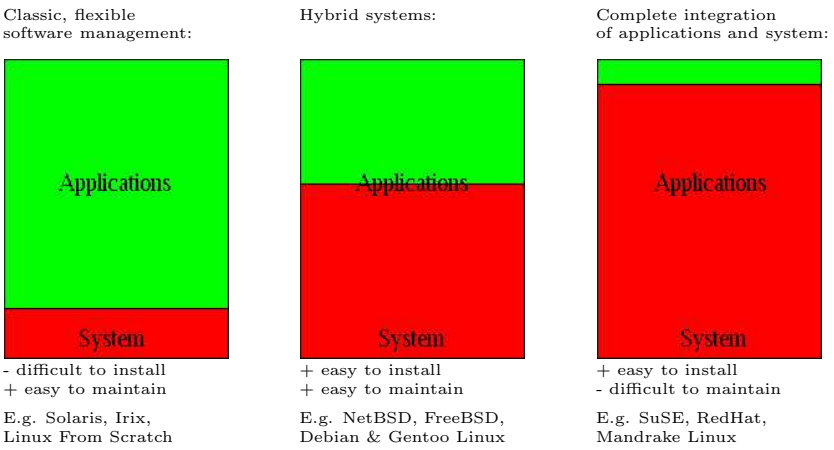


Figure 1: Degrees of integration between operating system and applications

3 A Cross-platform solution: pkgsrc

The pkgsrc system can be used for easy installation and updating of software packages. It is a source-based package management system which uses original source code for compiling, but also allows creation and installation of binary packages. The two major components are the management tools and the packages collection (pkgsrc).

The pkgsrc system handles dependencies between packages automatically with no user interaction. The system was originally ported from FreeBSD to NetBSD, and uses NetBSD as primary development platform today. In addition to NetBSD, pkgsrc was ported to IBM's AIX, BSDi/WindRiver's BSD/OS, Apple's Darwin, FreeBSD, SGI's Irix, various Linux distributions, OpenBSD, Sun's Solaris, and even Microsoft Windows in combination with Interix ("Services for Unix", SFU). Linux distributions known to work with pkgsrc are SuSE 9.0, Debian, ROOT Linux, Slackware, RedHat 8.1/9, Mandrake 9.2, and Bluewall Linux, the latter of which uses pkgsrc as its native packages system.

4 Getting started

In order to use pkgsrc, the following steps will be discussed:

- Download pkgsrc
- Install the bootstrap kit, either as binary or by compile via pkgsrc/bootstrap
- Install packages:

```
$ cd pkgsrc/www/mozilla
$ bmake install
```

4.1 Grabbing pkgsrc

The first step to use pkgsrc is to fetch it. This can either be done by downloading the tar-archive from ftp://ftp.NetBSD.org/pub/NetBSD/NetBSD-current/tar_files/pkgsrc.tar.gz, or by using anonymous CVS, following these steps:

```
$ cd $HOME/OS
$ env CVS_RSH=ssh \
  cvs -d anoncvs@anoncvs.NetBSD.org:/cvsroot \
  co pkgsrc
U pkgsrc/Makefile
U pkgsrc/Packages.txt
U pkgsrc/README
...
```

As pkgsrc is a fast moving system and frequent updates happen, CVS is better suited for later updating.

4.2 Bootstrapping with precompiled binaries

Before installing packages, the framework for installing needs to be bootstrapped first. This can be done by either using precompiled binaries of the framework, or by compiling manually.

Precompiled binaries are currently available for the following platforms:

Darwin 7.3.0/powerpc	IRIX 6.5/mips
Darwin 7.0/powerpc	IRIX64 6.5/mips
Darwin 6.6/powerpc	OpenBSD 3.2/i386
Debian Linux/i386	OpenBSD 3.5/i386
FreeBSD 3.5/i386	Slackware 8.1/i386
FreeBSD 5.1/i386	Slackware 9/i386
FreeBSD 5.2.1/i386	Solaris 8/sparc
Interix 3.5	Solaris 9/sparc
	Solaris 9/i386

4.3 Bootstrapping by compiling

An alternative to using precompiled bootstrap packages is compiling them from source, which also makes sure that the latest changes in the pkgsrc infrastructure are made available. Bootstrapping is done with the `pkgsrc/bootstrap/bootstrap` script.

Before starting, a decision needs to be made where packages should be placed into. Usually this is a place like `/usr/local`, `/usr/pkg` or `/opt` if the system is used site-wide. For demonstration purpose, the following examples assume that the system should be used without system (root) privileges, and be installed in the user's private home directory under `$HOME/OS`. The `pkgsrc` directory can be placed anywhere on the system, it is placed in `$HOME/OS/pkgsrc` here, and the operating system used here is SuSE 8.2 system.

The commands for bootstrapping then are:

```
$ cd pkgsrc/bootstrap
$ export MY_HOME=$HOME/OS/OS-‘uname -s‘
$ export LOCALBASE=${MY_HOME}/pkg
$ export PKG_DBDIR=${MY_HOME}/db/pkg
$ ./bootstrap \
?         --prefix=${LOCALBASE} \
?         --pkgdbdir=${PKG_DBDIR} \
?         --ignore-user-check
===> bootstrap command: ./bootstrap --prefix=/home/feyrer/OS/OS-Linux/pkg --pkgdbdir=/home/feyrer/OS/OS-Linux/db/pkg --ignore-user-check
===> bootstrap started: Wed Dec  8 14:42:23 CET 2004
Working directory is: work
===> running: /usr/bin/sed -e 's|@DEFAULT_INSTALL_MODE@|'0755'|' files/install-sh.in > work/install-sh
===> running: /bin/chmod +x work/install-sh
===> building as unprivileged user feyrer/bedienst
===> Building libnbcompat
===> running: /bin/sh work/install-sh -d -o feyrer -g bedienst work/libnbcompat
```

```
===> running: (cd work/libnbcompat; /bin/sh ./configure -C --prefix=/home/feyrer/OS/OS-Linux/pkg --sysconfdir=/home/feyrer/OS/OS-Linux/pkg/etc && make)
configure: creating cache config.cache
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking whether make sets $(MAKE)... yes
.....
..
.....
/usr/bin/install -c -m 444 linkfarm.cat1 /home3/bedienst/feyrer/OS/OS-Linux/pkg/man/cat1/linkfarm.0
/usr/bin/install -c -m 444 pkg_view.1 /home3/bedienst/feyrer/OS/OS-Linux/pkg/man/man1/pkg_view.1
/usr/bin/install -c -m 444 pkg_view.cat1 /home/feyrer/OS/OS-Linux/pkg/man/cat1/pkg_view.0
===> Installing packages(7) man page
===> running: /bin/sh work/install-sh -c -m 444 files/packages.cat7 /home/feyrer/OS/OS-Linux/pkg/man/cat7/packages.0

Please remember to add /home/feyrer/OS/OS-Linux/pkg/bin to your PATH environment variable
and /home/feyrer/OS/OS-Linux/pkg/man to your MANPATH environment variable, if necessary.

An example mk.conf file "work/mk.conf.example" with the settings you
provided to "bootstrap" has been created for you.
Please copy work/mk.conf.example to /home/feyrer/OS/OS-Linux/pkg/etc/mk.conf.

You can find extensive documentation of the NetBSD Packages Collection
in /home/feyrer/OS/pkgsrc/Packages.txt and packages(7).

Hopefully everything is now complete.
Thank you
===> bootstrap started: Wed Dec  8 14:44:09 CET 2004
===> bootstrap ended:   Wed Dec  8 14:55:52 CET 2004
$
```

After the pkgsrc framework is bootstrapped, paths need to be adjusted as printed at the end of the bootstrap process, and a call of the pkgsrc "`pkg_info`" command will show that there is already one package installed:

```
$ cd $HOME/OS/OS-‘uname -s‘/pkg
$ export PATH=‘pwd‘/bin:‘pwd‘/sbin:${PATH}
$ export PKG_DBDIR=$HOME/OS/OS-‘uname -s‘/db/pkg
$
$ pkg_info
digest-20021220     Message digest wrapper utility
```

4.4 Details on the bootstrapped system

The binaries installed by the bootstrap procedure provide the core functionality of the pkgsrc system:

```
% cd OS/OS-‘uname -s‘/pkg/
% ls bin sbin
bin:
bmake       cpio        digest      ftp
pax         tar

sbin:
linkfarm    pkg_add     pkg_create  pkg_info
mtree       pkg_admin   pkg_delete  pkg_view
```

Important commands to run later are the `pkg_*` programs as well as the `bmake` program. Manual pages for all these commands were installed as well, so documentation is readily available with the help of the Unix "`man`" command.

5 Using pkgsrc

After the bootstrap procedure has installed all the components needed to build and install packages, a first small package can be installed. **Beware!** Make sure that instead of "`make`" the BSD-

compatible "bmake" installed by the bootstrap procedure is being used. GNU make will definitely *not* work!

The commands to install the pkgsrc/misc/figlet package are:

```
$ export MAKECONF=‘pwd‘/pkgsrc_env_no-root # see below
$
$ cd $HOME/OS/pkgsrc
$ cd misc/figlet
$ bmake
...
$ bmake install
...
$
$ pkg_info
digest-20021220     Message digest wrapper utility
figlet-2.2.1nb2     Print text banners in fancy ASCII art characters
```

The first command ("`export MAKECONF=...`") adjust settings so software can be compiled and installed in a private place. The "`bmake`" and "`bmake install`" commands build the program and installs it into its target directory in `$HOME/OS`. The "`pkg_info`" command is used to verify that the package was installed properly, and the "`figlet`" command can be used now:

```
$ type figlet
/home/feyrer/OS/OS-Linux/pkg/bin/figlet
$
$ figlet Hello ‘uname -s‘
 _   _      _ _          _     _
| | | | ___| | | ___    | |   (_)_ __  _   ___  __
| |_| |/ _ \ | |/ _ \   | |   | | '_ \| | | \ \/ /
|  _  |  __/ | | (_) |  | |___| | | | | |_| |>  <
|_| |_|\___|_|_|\___/   |_____|_|_| |_|\__,_/_/\_\
```

5.1 More details on compiling and installing packages

The steps above illustrates the basic concept of installing software. This section gives a bit more information by providing information that is available during the build and install process.

The following output can be expected when building the figlet package:

```
% bmake
===> *** No /home/feyrer/OS/OS-Linux/../distfiles/pkg-vulnerabilities file found,
===> *** skipping vulnerability checks. To fix, install
===> *** the pkgsrc/security/audit-packages package and run
===> *** '/home/feyrer/OS/OS-Linux/pkg/sbin/download-vulnerability-list'.
=> Checksum OK for figlet221.tar.gz.
work.i386 -> /home/feyrer/OS/OS-Linux/tmp/misc/figlet/work.i386
===> Extracting for figlet-2.2.1nb2
===> Patching for figlet-2.2.1nb2
===> Applying pkgsrc patches for figlet-2.2.1nb2
===> Overriding tools for figlet-2.2.1nb2
===> Configuring for figlet-2.2.1nb2
===> Building for figlet-2.2.1nb2
gcc -O2 -DDEFAULTFONTDIR=\"/home/feyrer/OS/OS-Linux/pkg/share/figlet\"  -DDEFAULTFONTFILE=\"standard.flf\"  figlet.c zipi
chmod a+x figlet
gcc -O2 -o chkfont chkfont.c
%
```

After compilation, the binaries are installed in a second step:

```
% bmake install
===> Installing for figlet-2.2.1nb2
===> Becoming root@rfhinf032 to install figlet.
Warning: not superuser, can't run mtree.
Become root and try again to ensure correct permissions.
install -d -o feyrer -g bedienst -m 755 /home/feyrer/OS/OS-Linux/pkg/man/man6
mkdir -p /home/feyrer/OS/OS-Linux/pkg/share/figlet
cp figlet /home/feyrer/OS/OS-Linux/pkg/bin
cp chkfont /home/feyrer/OS/OS-Linux/pkg/bin
chmod 555 figlist showfigfonts
cp figlist /home/feyrer/OS/OS-Linux/pkg/bin
cp showfigfonts /home/feyrer/OS/OS-Linux/pkg/bin
cp fonts/*.flf /home/feyrer/OS/OS-Linux/pkg/share/figlet
cp fonts/*.flc /home/feyrer/OS/OS-Linux/pkg/share/figlet
cp figlet.6 /home/feyrer/OS/OS-Linux/pkg/man/man6
===> Registering installation for figlet-2.2.1nb2
$
```

5.2 Compiling as non-root

Normally, installation of software needs system (root) privileges, to install software into special directories that are not writable by normal users. Pkgsrc can be used without these system privileges to quite some extent. To do so, a number of variables need to be set, and the `$MAKECONF` environment variable needs to be pointed at that file:

```
$ export MAKECONF=‘pwd‘/pkgsrc_env_no-root
$ ls -la $MAKECONF
-rw-rw-r--  1 feyrer  bedienst  816 Oct  6 04:46 /home/feyrer/OS/pkgsrc_env_no-root
```

The full version of the `pkgsrc_env_no-root` can be found in appendix A.

5.3 Behind the scenes

In the above example, a software package was installed in two separate steps with two separate commands, "`bmake`" and "`bmake install`". It can have been done in one step with just "`bmake install`", and still, building and all the other steps needed first will be performed. If needed, the steps can be ran manually as well, and the following list shows the commands for manual execution as well as the action performed:

1. `bmake fetch`: Download sources
2. `bmake checksum`: Ensure integrity of sources
3. `bmake install-depends`: Install required packages
4. `bmake extract`: Unpack sources
5. `bmake patch`: Apply patches kept in pkgsrc
6. `bmake configure`: Configure
7. `bmake build`: Compile
8. `bmake install`: Install and register package (for pkg_info(1), pkg_delete(), etc.)

Other targets that may be useful are:

- `bmake package`: Create binary package for pkg_add(8)
- `bmake clean`: Remove work directory

- `bmake deinstall`: Deinstall package
- `bmake replace`: Replace installed package with new version
- `bmake update`: Rebuild package and all dependencies

These lists are by no means complete. Please see the pkgsrc guide in pkgsrc/doc/pkgsrc.txt and the packages(7) manpage for more information.

6 Overview of available packages

Currently, pkgsrc itself contains almost 5200 packages, and the SourceForge pkgsrc-wip ("Work in Progress") project contains almost 1000 more packages. The packages in pkgsrc are organized in categories, with one directory per category, and package directories in the category directory. For example, the Mozilla package can be found in `pkgsrc/www/mozilla`, KDE3 is in `pkgsrc/meta-pkgs/kde3` and so on.

Here is an example listing all existing categories:

```
$ cd .../pkgsrc/
$ ls
CVS               databases         lang              pkglocate
Makefile          devel             licenses          pkgtools
Packages.txt      distfiles         mail              print
README            doc               math              regress
archivers         editors           mbone             security
audio             emulators         meta-pkgs         shells
benchmarks        finance           misc              sysutils
biology           fonts             mk                templates
bootstrap         games             multimedia        textproc
cad               geography         net               time
chat              graphics          news              wm
comms             ham               packages          www
converters        inputmethod       parallel          x11
cross
```

As an example of the WWW category, here is a fraction of the packages contained in it:

```
$ cd .../pkgsrc
$ ls www
CVS               cadaver           jakarta-servletap p5-Apache-Test
Makefile          calamaris         jakarta-tomcat    p5-Apache-ePerl
Mosaic            cgic              jakarta-tomcat4   p5-CGI
SpeedyCGI         cgicc             jsdk20            p5-CGI-Applicatio
adzap             cgilib            jssi              p5-CGI-FastTempla
amaya             checkbot          kannel            p5-CGI-FormBuilde
analog            chimera           kdewebdev3        p5-CGI-Kwiki
ap-Embperl        clearsilver       kimagemapeditor   p5-CGI-Minimal
ap-access-referer cocoon            lhs               p5-CGI-Session
ap-aolserver      communicator      libghttp          p5-CGI_Lite
ap-auth-cookie    cronolog          libgtkhtml        p5-ExtUtils-XSBui
ap-auth-ldap      curl              libwww            p5-FCGI
ap-auth-mysql     cvsweb            liferea           p5-HTML-Clean
ap-auth-pam       dillo             links             p5-HTML-FillInFor
ap-auth-pgsql     drivel            links-gui         p5-HTML-FixEntiti
ap-auth-postgresq elinks            lynx              p5-HTML-Format
ap-auth-script    elinks04          mMosaic           p5-HTML-Mason
ap-bandwidth      emacs-w3m         make_album        p5-HTML-Parser
...
```

7 Conclusion

This article contains a small and short overview about software management, showing the importance of systems to assist installation of software packages in systems that use a large number of modules as can be found in Open Source systems today, and introduces the pkgsrc system which can be used on a variety of hardware and operating system platforms to install and maintain software.

More information about internals of the system, dependency handling etc. would be beyond the scope of this document, but can be found in the pkgsrc guide at `pkgsrc/doc/pkgsrc.txt` and on the websites of the pkgsrc and the NetBSD projects, see:

http://www.pkgsrc/org/

http://www.NetBSD.org/Documentation/pkgsrc/

A pkgsrc_env_no-root

```
# make(1) include file for NetBSD pkgsrc as non-root
#
# Usage:
# env MAKECONF=/path/to/pkgsrc_env make ...
#
# (c) Copyright 2003, 2004 Hubert Feyrer <hubert@feyrer.de>
#

MY_NAME!=       whoami
MY_GROUP!=      groups | sed 's/ .*$$//'
MY_OS!=         uname -s
MY_HOME=        ${HOME}/OS/OS-${MY_OS}

BINOWN=         ${MY_NAME}
BINGRP=         ${MY_GROUP}
SHAREOWN=       ${MY_NAME}
SHAREGRP=       ${MY_GROUP}
MANOWN=         ${MY_NAME}
MANGRP=         ${MY_GROUP}

WRKOBJDIR=      ${MY_HOME}/tmp
PKG_DBDIR=      ${MY_HOME}/db/pkg
OBJMACHINE=     1

DISTDIR=        ${MY_HOME}/../distfiles
PACKAGES=       ${MY_HOME}/packages

# X needs xpkgwedge installed!
LOCALBASE=      ${MY_HOME}/pkg
VARBASE=        ${MY_HOME}/var

SU_CMD=         /bin/sh -c
FETCH_CMD=      ${LOCALBASE}/bin/ftp
PAX=            ${LOCALBASE}/bin/pax
CHOWN=          true
CHGRP=          true
BINMODE=        755                     # for Solaris strip(1)

# For apache (needs patch to use VARDIR):
APACHE_USER=    ${MY_NAME}
APACHE_GROUP=   ${MY_GROUP}
```

The latest version of this file can be found at http://www.feyrer.de/OS/pkgsrc_env_no-root!