Pkgsrc binary packages management with pkgin

Emile “iMil” Heitor

FOSDEM 2012

January 31, 2012
Who am I

- Emile “iMil” Heitor, french, born on 1974
- 1st OSS contribution on 1995
- Founder of the GCU-Squad
- NetBSD user since 1997
- NetBSD developer since 2009
- CTO of NBS-System, a french OSS Internet hosting company
How and when it all started

- Debian GNU/Linux user, don’t lie, `apt` is great
- Fed up with binary packages upgrades systems on `NetBSD`
- First discussion about a binary package manager on March 2 2009
- First commit on Mar 9 10:34:34 2009 UTC
- Previous name was “pkg_dry”
Dependencies and languages

- Written using the C language
- Build dependencies
  - libarchive
  - libfetch
  - pkg_install
- Build/run dependencies: SQLite3
- Around 3500 lines of code (95% C)
What is and what is not pkgin

- *pkgin* is **not** a pkgsrc (as in source) manager
- *pkgin* is a **binary** package manager
- *pkgin* **only handles** dependencies
- *pkgin* tries to mimic *apt*’s behaviour
- Calls *pkg_install* for actual installation / removal
- Relies on *pkg_info* for metadatas
Supported (and tested) platforms
Supported (and tested) platforms

NetBSD

DragonFly BSD
Supported (and tested) platforms

![NetBSD](https://www.netbsd.org/badges/logo-lg.png)

![DragonFly BSD](https://www.dragonflybsd.org/dragonfly.png)

- Emile “iMil” Heitor
- Binary package management with pkgin
Supported (and tested) platforms

- NetBSD
- DragonFly BSD
- Solaris
Supported (and tested) platforms

- NetBSD
- DragonFlyBSD
- solaris
- pkgin

Emile “iMil” Heitor
Binary package management with pkgin
Supported (and tested) platforms

- NetBSD
- DragonFly BSD
- solaris
- Linux
A pkg_summary(5) file is generated for every bulk build
A `pkg_summary(5)` file is generated for every *bulk build*

<table>
<thead>
<tr>
<th><strong>pkg_summary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>PKGNAME=pkgfind-20050804</td>
</tr>
<tr>
<td>COMMENT=Find packages by package name in pkgsrc</td>
</tr>
<tr>
<td>SIZE_PKG=10591</td>
</tr>
<tr>
<td>BUILD_DATE=2011-01-16 23:46:59 +0000</td>
</tr>
<tr>
<td>CATEGORIES=pkgtools</td>
</tr>
<tr>
<td>MACHINE_ARCH=i386</td>
</tr>
<tr>
<td>OPSYS=NetBSD</td>
</tr>
<tr>
<td>OS_VERSION=5.0.2</td>
</tr>
<tr>
<td>PKGPATH=pkgtools/pkgfind</td>
</tr>
<tr>
<td>PKGTOOLS_VERSION=20100204</td>
</tr>
<tr>
<td>REQUIRES=/usr/lib/libc.so.12</td>
</tr>
<tr>
<td>FILE_NAME=pkgfind-20050804.tgz</td>
</tr>
<tr>
<td>FILE_SIZE=6703</td>
</tr>
<tr>
<td>DESCRIPTION=pkgfind can find packages in pkgsrc. It tries to find packages which matches a keyword in the package name.</td>
</tr>
</tbody>
</table>
Our choices

- Parsing plain text... damn slow
- Loading `pkg_sumary` into memory... overkill
- Using `bdb`, `cdb`... where reasonable options
- **SQLite**!
### Structure of the SQLite database

**SQLite structure**

```sql
sqlite> .tables
LOCAL_CONFLICTS  LOCAL_PROVIDES  REMOTE_CONFLICTS  REMOTE_PROVIDES
LOCAL_DEPS        LOCALQUIRES     REMOTE_DEPS        REMOTEQUIRES
LOCAL_PKG         PKGDB           REMOTE_PKG         REPOS

sqlite> select pkg_id,pkgname,pkgvers,pkgpath from local_pkg where pkgname='nginx';

<table>
<thead>
<tr>
<th>PKG_ID</th>
<th>PKGNAME</th>
<th>PKGVERS</th>
<th>PKGPATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>nginx</td>
<td>1.0.4</td>
<td>www/nginx</td>
</tr>
</tbody>
</table>

sqlite> select * from local_deps limit 5;

<table>
<thead>
<tr>
<th>LOCAL_DEPS_ID</th>
<th>PKG_ID</th>
<th>LOCAL_DEPS_PKGNAME</th>
<th>LOCAL_DEPS_DEWEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>15258</td>
<td>1</td>
<td>jpeg</td>
<td>jpeg&gt;=8nb1</td>
</tr>
<tr>
<td>15259</td>
<td>5</td>
<td>kpathsea</td>
<td>kpathsea&gt;=6.0.0</td>
</tr>
<tr>
<td>15260</td>
<td>8</td>
<td>lcms</td>
<td>lcms&gt;=1.12nb2</td>
</tr>
<tr>
<td>15261</td>
<td>8</td>
<td>jpeg</td>
<td>jpeg&gt;=8nb1</td>
</tr>
<tr>
<td>15262</td>
<td>8</td>
<td>png</td>
<td>png&gt;=1.5.0</td>
</tr>
</tbody>
</table>
```
Retrieving what matters

Listing package dependencies

dependencies query (pkgindb_queries.c)

```
const char DIRECT_DEPS[] = /* prefer higher version */
   "SELECT REMOTE_DEPS_DEWEY, REMOTE_DEPS_PKGNAME 
   "FROM REMOTE_DEPS WHERE PKG_ID = "
   "(SELECT PKG_ID FROM REMOTE_PKG WHERE PKGNAME = '%s' 
   "ORDER BY FULLPKGNAME DESC LIMIT 1);"
```

Example

1st level “kdenlive” dependencies

```
sqlite> SELECT REMOTE_DEPS_DEWEY, REMOTE_DEPS_PKGNAME FROM REMOTE_DEPS WHERE PKG_ID = \n   (SELECT PKG_ID FROM REMOTE_PKG WHERE PKGNAME = 'kdenlive' ORDER BY FULLPKGNAME DESC LIMIT 1);
desktop-file-utils>=0.10nb1|desktop-file-utils
dvdauthor>=0.6.18|dvdauthor
kdelibs4>=4.5.5nb3|kdelibs4
mlt>=0.5.10|mlt
qt4-libs>=4.7.3nb1|qt4-libs
qt4-qdbus>=4.7.2nb1|qt4-qdbus
```
Inside pkgin’s brain

The main objective is to provide a reliable and evolutive packages map

- Dependencies discovery
  - Retrieve 1st level dependencies
  - loop until we have a “full dependency tree”

- Impact calculation
  - Either for package installation, upgrade or removal
  - For each selected package, retrieve its “full dep tree”
  - Match it against what we have and what we would have
  - Match it against the conflict table

- Actions ordering
  - Installation: from the lesser dependency deepness
  - Removal: from the higher dependency deepness
  - Upgrade is nothing more that removal followed by installation
pkgin 0.5 features

- List, search, available packages
- Package + dependencies installation
- Package + dependencies upgrade (partial or full)
- Package + dependencies removal
- Various dependencies informations (*sd, sfd, srd, prov, req*)
- Unneeded packages removal (*autoremove*)
- Export / import packages list
Typical usage

- foo# pkgin up
- foo# pkgin in texmaker
- foo# pkgin ex > my-pkg-list
- bar# pkgin im my-pkg-list
- foo# pkgin rm kdenlive
- foo# pkgin ar

Yay, a new pkgsrch release!

- foo# pkgin up
- foo# pkgin fug
pkgin 0.6+ features

- More testing features
- User Interface using PackageKit, WIP (Sylvain Mora)
- More SQL, less C (work in progress with Baptiste from FreeBSD)
- Fix the multi-repository feature (don’t use it at home kids!)
pkgin (and NetBSD) needs you.
Questions?