Cross-Platform Multi-Instance Unix Software Packaging

package once — use everywhere !

The best way to **predict** the **future** is to **invent** it.
— *Alan Kay*

Only those who attempt the **absurd** can achieve the **impossible**.
— *Unknown*

Ralf S. Engelschall
rse@openpkg.org

2.5 2006-03-19

Part I: Name Of The Game

What is the Problem?
Why Packaging at all?
Why Cross-Platform?

There are two types of **people** in this world, **good** and **bad**. The good **sleep better**, but the bad seem to **enjoy** the **waking hours** much more.

— *Woody Allen*

What is the Problem? (1)

- **Cross-Platform:**
  How to manage different Unix platforms without having to deal with different vendor facilities?
- **Trust:**
  How to trust any vendor unless their whole project workflow and results are public and transparent?
- **Organizational Separation:**
  How to achieve a clean responsibility separation on servers between System Administrators and Application Administrators/Developers?
- **Bleeding Edge:**
  How to use a software just a few hours after it was released by the vendor?
- **Package Variants:**
  How to deploy multiple variants (build-time options) of a software with an arbitrary vendor packaging facility?
- **Multiple Instances:**
  How to use staging installations without having to buy additional dedicated servers?

Understanding a **problem** is knowing why it is hard to solve it, and why the most straightforward approaches won't work. — *Karl Popper*

What is the Problem? (2)

- **Sane Build Environment:**
  How to build packages in a sane and well-defined environment?
- **Unprivileged Packaging:**
  How to build binary packages without write access to the target filesystem area?
- **Unprivileged Deployment:**
  How to use a software packaging facility in a fully unprivileged deployment environment?
- **Building from Source:**
  How to reproduce a software installation from pristine vendor sources directly on the end-user target machine?
- **Conciseness/Cleanness:**
  How to trust the resulting binary packages if the packaging itself is not already known to be maximum concise, clean and reviewable?
- **Safe Environment:**
  How to make sure that own solutions are future safe by not being too tied to a particular underlying operating system?

It's not enough to be a great **programmer**;
you have to find a great **problem**.
— Charles Simonyi

Why Packaging at all? (1)

- **Reproducibility:**
  Packaging allows to really reproduce the resulting software installation.
- **Filesystem Intrusion:**
  Packaging allows to exactly know what files form a piece of software. Later removal is possible without any residual files.
- **Scalability:**
  Packaging allows software deployment to be independent on the required number of deployments.
- **Unification:**
  Packaging unifies individual approaches across application vendors to simplify administration.
- **Problem Focusing:**
  Packaging allows to focus on the problem (deployment and configuration) instead of having to fight (again and again) against the porting and building of vendor software.
- **Cost Reduction:**
  Packaging reduces costs by no longer requiring experts for boring deployment tasks. Instead their expertise can be used for service improvements.

"**Reuse** an expert's code" is the right advice for most people. But it's **useless** advice for the experts writing the code **in the first place**. — *Dan J. Bernstein*

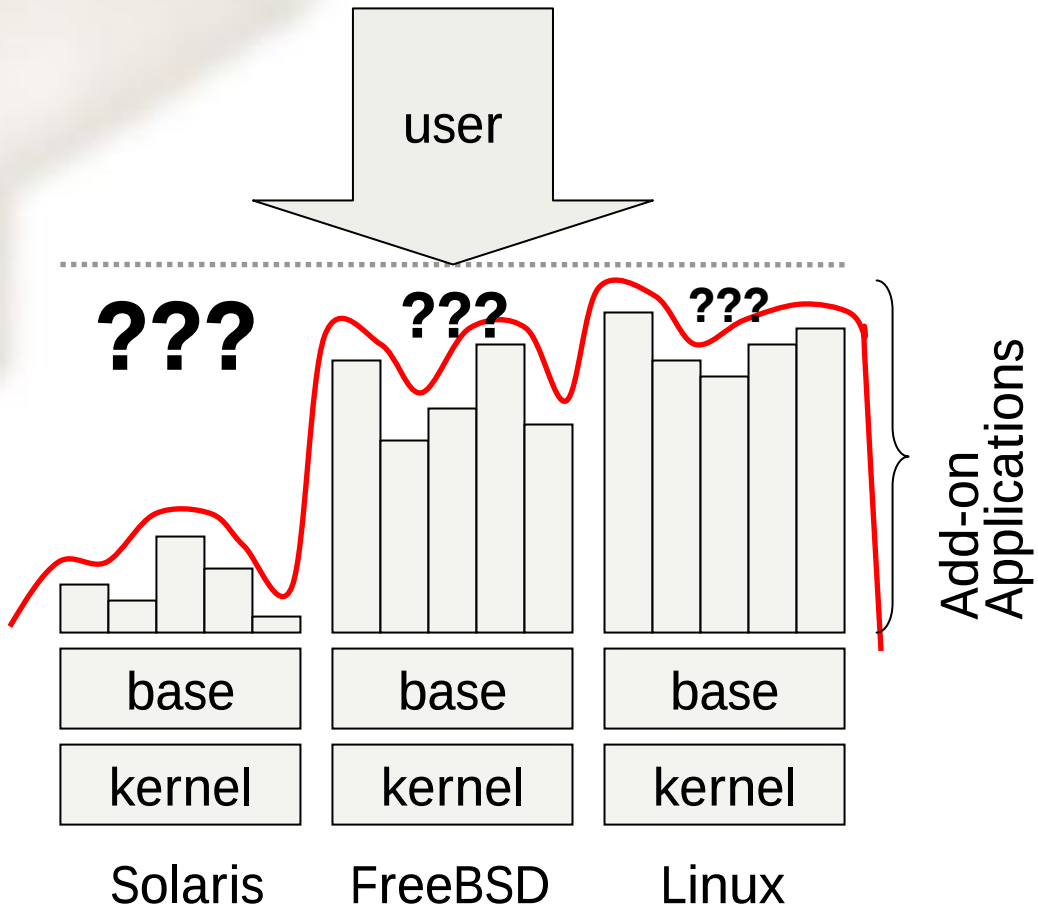
Why Packaging at all? (2)

- **Built-In Experience:** Packaging combines vendor applications with pre-configuration and packager knowledge to create optimum total result.
- **Knowledge Consolidation:** Packaging allows central consolidation of knowledge.
- **Patch Maintenance:** Packaging allows you to keep pristine vendor sources and patches separate without loosing seamless integration.
- **Annotations:** Packaging annotates vendor applications with useful meta information for administration.
- **Querying Information:** Packaging allows to reasonably query information about the application installation.
- **Safe Upgrade Path:** Packaging allows a guaranteed upgrade path for software during the whole life of a system.
- **Integrity Verification:** Packages can be signed and their content integrity can be verified.

The reasonable man **adapts** himself **to the world**; the unreasonable one persists in trying to **adapt** the world **to himself**. Therefore all progress depends on the unreasonable man. *— George Bernard Shaw*

Why Cross-Platform? (1)
The Mountain Problem

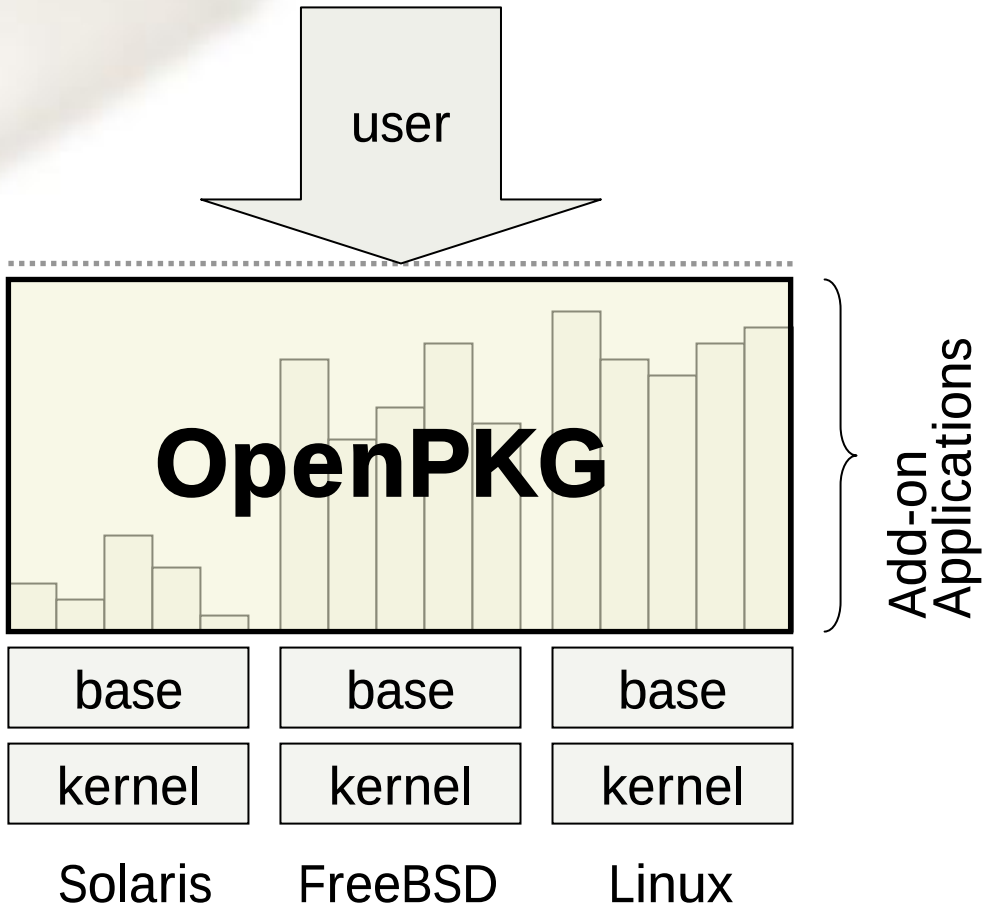
- Different flavors of Unix operating systems have to be used and cannot be avoided.
- Differences in vendor supplied add-on applications:
  - Total number of applications.
  - Third-party application versions.
  - Used filesystem layout.
  - Particular chosen build-time options.
  - Amount of pre-configuration.
- Administrators have to know how to manage **n** different platforms.



There's a lesson to be **learned** from this but I'll be damned if I know what it is.
— Al Bundy

Why Cross-Platform? (2)
The OpenPKG Solution

- Different flavors of Unix operating systems are still being used because cannot be avoided.
- Vendor supplied add-on applications are deinstalled or at least shadowed by OpenPKG.
- OpenPKG is a maximum independent layer on top of the operating system.
- All add-on applications are provided as cross-platform packages by OpenPKG.
- Administrators now just have to know how to manage **1** unified platform.



The software said it requires **Solaris 9** or better, so I installed **OpenPKG**...

Part II: The Solution

The Solution: Overview
The Solution: Design Goals
Platform Availability
Platform Classification
Package Classification
Packaging Approaches

The **solution** of this problem
is **left** as an exercise **to the reader**.

The Solution: Marketing Style

- OpenPKG — The Cross-Platform Multi-Instance Unix Software Packaging Facility.
- Much valued by IT decision makers and beloved by Unix system administrators, OpenPKG is the world leading instrument for deployment and maintenance of Open Source software when administration crosses Unix platform boundaries.
- The unique OpenPKG architecture leverages proven technologies like Red Hat Package Manager (RPM) and OSSP and GNU components to establish a unified software administration environment, independent of the underlying Unix operating system.

Software is like **sex**;
it's better when it's **free**.
— *Linus Torvalds*

The Solution: Technology Style

LOAD "**OPENPKG**",8,1

- a cross-platform packaging facility for Unix software.
- based on a ported, cleaned up and extended version of the popular Red Hat Package Manager (RPM 4.2).
- a fully self-contained packaging facility which is maximum independent of underlying operating system.
- minimum intrusion during linkage into the underlying operating system (just 6 connection points).
- very complete, i.e., it currently provides already over 880 packaged applications.
- a mature technology now in production use since 4 years.
- freely available to anyone as Open Source under a MIT-style distribution license.
- releases are provided three times per year and the last two releases are fully covered with security updates.

The Solution: Design Goals

- Design Goal 1: **Packaging at all** (keywords: complexity, removability, reproducability, scalability)
- Design Goal 2: **Cross-Platform** (keywords: inherent constraints, flexibility, cost reducation)
- Design Goal 3: **Multiple Instances** (keywords: complexity, flexibility, utilization, evaluation, staging)
- Design Goal 4: **Out-of-the-Box Configuration** (keywords: minimum default, maximum usability, experience bundling)
- Design Goal 5: **Accuracy & Conciseness** (keywords: artwork, human friendliness, maintainability)
- Design Goal 6: **Covering Essentials Only** (keywords: “best of”, quality not quantity, major Unix flavors)
- Design Goal 7: **Open Source Licensing** (keywords: “free as in freedom, not as in free beer”)

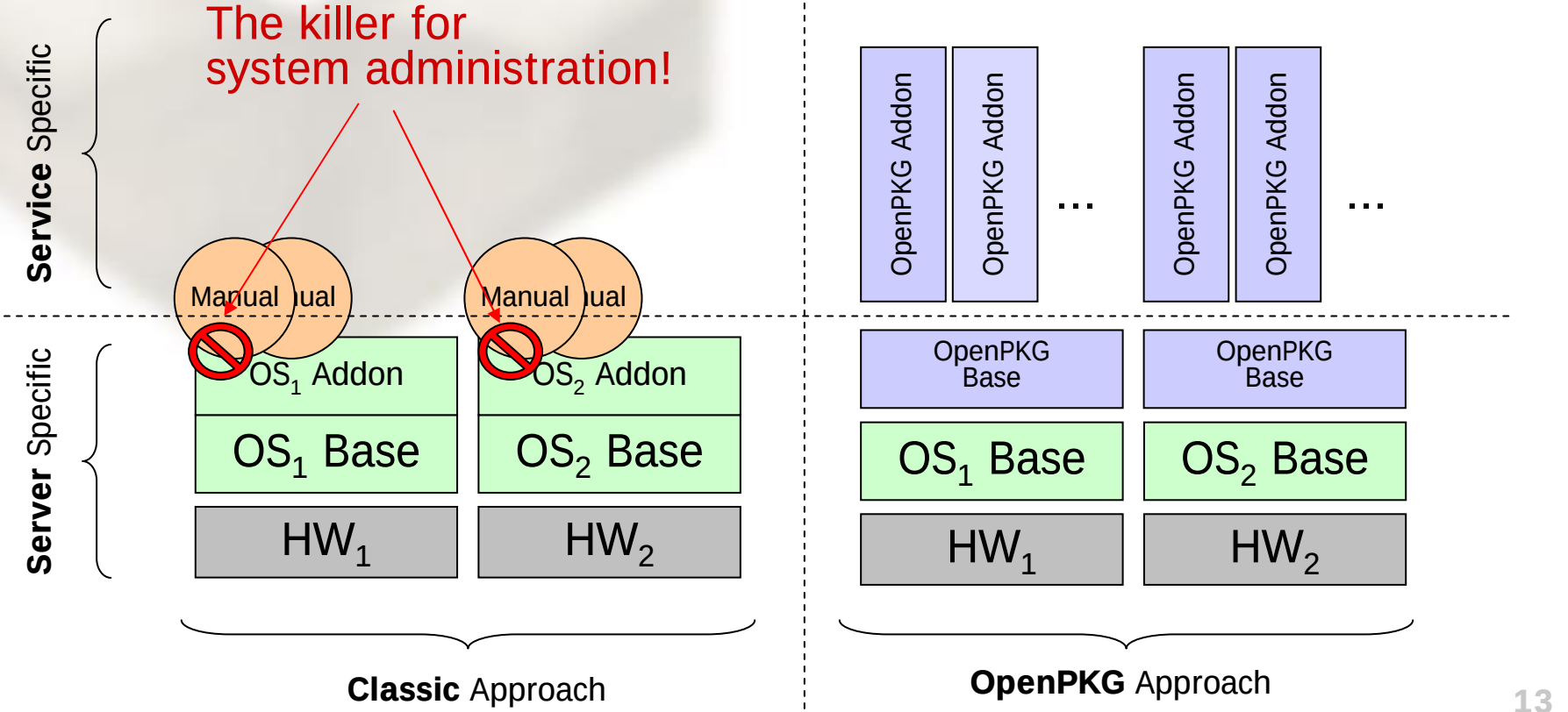
I'd like to thank all the little people who **helped** make this possible, but I can't, because I did it all **myself**.
— *Herman Monster*

Good design means **less** design.
Design must serve users, not try to fool them.
— *Dieter Rams, Chief Designer, BRAUN*

The Solution: “Big Picture”

We the **unwilling**,
led by the **unknowing**,
are doing the **impossible**.
— *Larry Wall*

- In the Classic approach, addon OS vendor packages plus manually installed software provide services.
- In the OpenPKG approach, an OpenPKG Base instance extends the OS Base installation and dedicated OpenPKG instances provide services.



Platform Availability

- OpenPKG is officially available for mainly 3 Unix platform technologies:
  - FreeBSD
  - GNU/Linux
  - Sun Solaris
- OpenPKG is officially available for 21 particular platform products (as of OpenPKG 2.3)
- For every release, all packages are built on all platforms.

- **FreeBSD**
  - FreeBSD 4.11 (iX86)
  - FreeBSD 5.3 (iX86)
  - FreeBSD 5.3 (SPARC64)
  - FreeBSD 5.3 (IA64)
  - FreeBSD 6.0 (iX86)
- **GNU/Linux**
  - Debian GNU/Linux 3.0 (iX86)
  - Debian GNU/Linux 3.1-PRE (iX86)
  - RedHat Enterprise Linux 3 (iX86)
  - Fedora Core 3 (iX86)
  - SuSE Enterprise Linux 9 (iX86)
  - SuSE Linux 9.2 (iX86)
  - Gentoo Linux 1.6.9 (iX86)
  - Mandrake Linux 10.1 (iX86)
- **Sun Solaris**
  - Sun Solaris 8 (SPARC64)
  - Sun Solaris 9 (iX86)
  - Sun Solaris 9 (SPARC64)
  - Sun Solaris 10 (ix86)
  - Sun Solaris 10 (SPARC)
- **Others**
  - NetBSD 2.0 (iX86)
  - HP HP-UX 11.11i (HPPA)
  - Apple Darwin 7.8 (PPC)

Platform Classification

- OpenPKG platforms are classified into 5 categories:
  - deprecated
  - obsolete
  - supported
  - tentative
  - forecasted

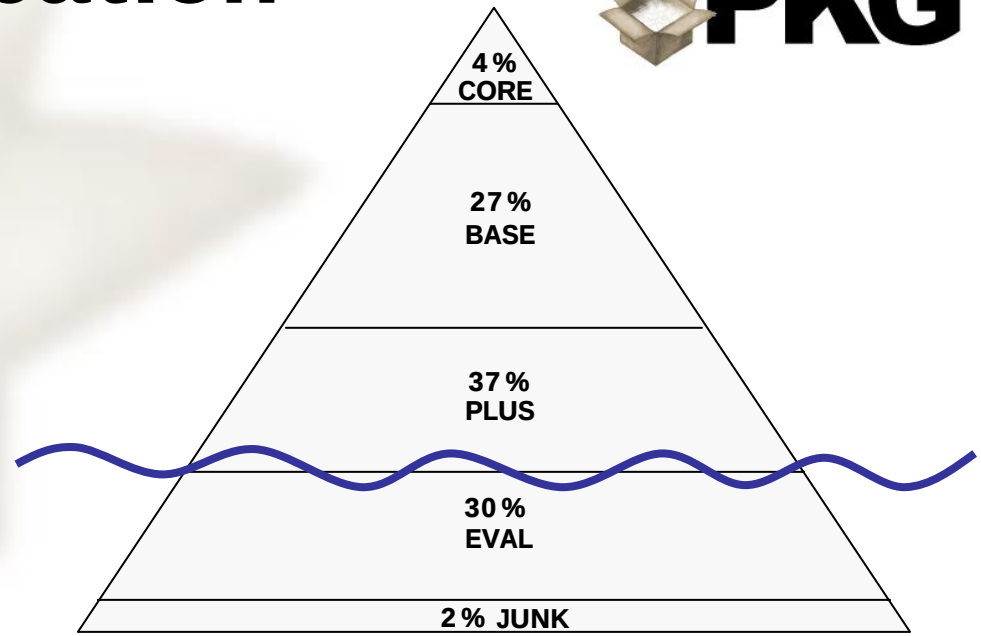
- As the name implies, only “supported” platforms are really officially supported!
- Availability on “obsolete” platforms is still provided for convenience reasons only.
- Availability on “tentative” platforms is already provided for early adopter and testing reasons.

deprecated	obsolete	supported	tentative	forecasted
unixware tru64		ix86-freebsd5.3 ix86-freebsd4.11 ix86-debian3.0 ix86-fedora3 ix86-rhel3 ix86-suse9.2 ix86-suse9 sparc64-solaris8 ix86-solaris9 sparc64-solaris9 ix86-solaris10 sparc64-solaris10	ix86-freebsd6.0 ia64-freebsd5.3 sparc64-freebsd5.3 ix86-gentoo1.6.9 ix86-debian3.1 ix86-mandrake10.1 ix86-netbsd2.0 ppc-darwin7.8.0 hppa-hpux11.11	aix irix

Reporter: What do you think of **Western Civilization**?
Ghandhi: I think, it would be a good idea.

Package Classification

- OpenPKG packages are classified into 5 categories:
  - CORE, BASE, PLUS
  - EVAL, JUNK
- Classification of a package depends on:
  - CORE, BASE: decision by principal architect.
  - PLUS: decision by principal architect and package status.
  - EVAL, JUNK: package status.



- The upper half of the “iceberg” (CORE, BASE and PLUS) make up the official releases.
- PLUS packages are going in and out as neccessary during release engineering.

	CORE	BASE	PLUS	EVAL	JUNK
Packaging Completed	X	X	X	X	-
Build-Time Tests Successful	X	X	X	?	-
Run-Time Tests Successful	X	X	(X)	?	-
Release: Showstopper	X	X	-	-	-
Release: Source Package	X	X	X	-	-
Release: Binary Package	X	X	-	-	-
Security Engineering	X	X	(X)	-	-

"If builders built **buildings** the way programmers wrote **programs**, then the first **woodpecker** that came along would **destroy civilization**."
— Weinberg's second law

Packaging Approaches: Source vs. Binary

- There are two fundamentally different approaches for packaging-based software distributions:
  - providing source packages containing the vendor sources plus instructions for automated build and installation.
  - providing binary packages containing the final installation files only.
- Most packaging facilities support both approaches (including RPM), although often not equally well.
- Both approaches have each their pros and cons, nevertheless all software distributions focus on one of them.

Beware of **programmers** who carry **screwdrivers**.
— *Leonard Brandwein*

- OpenPKG is focused on source packages because of the proofed success of reproducably building from prestine vendor sources.
- In OpenPKG, binary packages are just an intermediate temporary result (or used for bootstrapping and emergency situations) only.

	source package	**binary package**
distribution size	☺☺☺	☹☹☹
package size	😐	☺
package dependencies	☹☹	😐
installation reproducability	☹	☺☺☺
installation run-time stability	☺☺	☹
installation system alignment	☺☺☺	☹
installation time	☹☹	☺☺

Part III: About Project

Project Roots
Project Roadmap
Engineering Phases
Who's Who?

A **distributed system** is one on which I cannot get any work done, because a machine I have never heard of has crashed.

— *Leslie Lamport*

About Project: The Roots

Premature **optimization** is the root of all evil.
— D. E. Knuth

- OpenPKG dates back in concept to 1992 when Ralf S. Engelschall (RSE) developed his Build'n'Play (BnP) and GenOPT at sd&m (sdm.de).
- BnP was a Perl based build environment for easy installation of Unix software on FreeBSD and Sun Solaris.
- GenOPT was a complex shell script which allowed to link the locally installed software into a global access layer.
- When in November 2000 RSE went to Cable & Wireless (cw.com) the BnP/GenOPT approach was not sufficient and a more complete and integrated solution was aspired.
- In-depth evaluation of major packaging facilities showed that none was able to fulfill all(!) requirements.
- Fortunately, RPM proved to be the most balanced solution, because it covers at least 80% of every(!) requirement.
- RPM was chosen, ported to more non-RedHat-Linux platforms and embedded into a elaborate bootstrapping procedure.
- On top of this, the first dozen RPM packages were developed by converting the BnP Perl/sh scripts to RPM Bash scripts.
- OpenPKG 0.9 was born!

OpenPKG RPM: PM Requirements

Engineering does not require **science**. Science helps a lot, but people built perfectly good **brick walls** long before they knew why **cement** works. — *Alan Cox*

- The OpenPKG project had the following major requirements to the Package Manager:
  - The PM has to be maximum portable to all major Unix platforms and require the minimum on other software.
  - The PM has to cover the full life-cycle of a package, starting from tracking the vendor sources to the residue-free deinstallation of the installed package.
  - The PM has to be flexible enough to be easily extensible with OpenPKG extensions.
  - The PM has to be driven with a single all-in-one package specification and through a integrated command line interface.
- The OpenPKG project evaluated (in Nov. 2000) the following PM implementations:
  - FreeBSD 4.x Ports/pkg_xxx
  - Debian 2.2 dpkg/Apt
  - Sun Solaris 8 pkgxxx
  - RedHat RPM 4.0
- OpenPKG chose RedHat RPM because
  - it covered already 80% of all OpenPKG requirements.
  - the remaining 20% were added easily by OpenPKG.
- As of OpenPKG 2.3, the RPM 4.2.1 extensions are about
  - 9500 LoC shell extensions
  - 5000 LoC C patches
  - 450 LoC macro additions
  - 150 LoC CLI aliases

About Project: The Roadmap

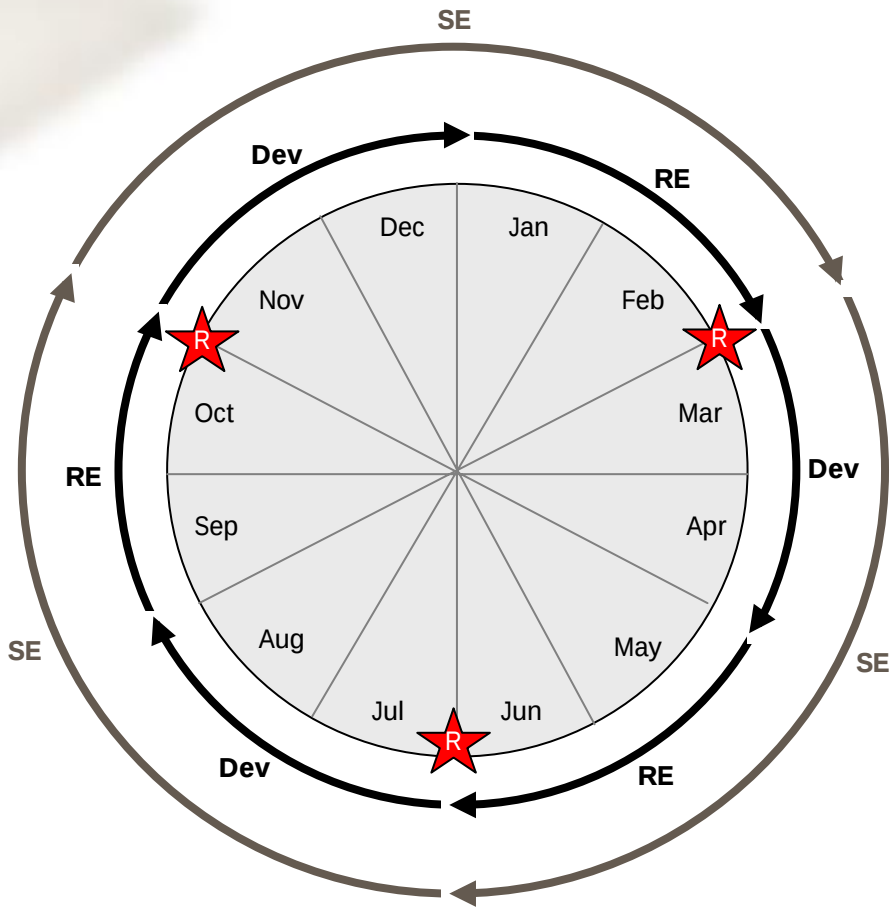
- As of April 2005, OpenPKG already went through 8 official releases since 2001.
- Release Engineering is performed within 4-6 weeks every 4 months in order to ship 3 releases per year.
- Security Engineering is performed constantly for the last 2 releases.
- OpenPKG-CURRENT is constantly updated on a bi-daily basis with the latest vendor versions.

Date	Milestone
Nov-2000	OpenPKG project kick-off
Apr-2001	OpenPKG 0.9, C&W deployment
Jan-2002	**OpenPKG 1.0**
Mar-2002	feature: {s,m,r,n}{usr,grp}
Jun-2002	feature: sane build environment
Aug-2002	**OpenPKG 1.1**
Nov-2002	feature: RDF, openpkg-tool, FSL
Dec-2002	feature: %option
Jan-2003	**OpenPKG 1.2**
Apr-2003	feature: GCC 3.3, RC work-off
Aug-2003	**OpenPKG 1.3**
Oct-2003	feature: RPM 4.2.1, platform
Jan-2003	feature: UUID, %track/Class, tag
Feb-2004	**OpenPKG 2.0**
May-2004	feature: OpenPKG Tool Chain, gcc 3.4
Jun-2004	**OpenPKG 2.1**
Oct-2004	**OpenPKG 2.2**
Feb-2005	**OpenPKG 2.3**
Mar-2005	OpenPKG Foundation e.V., SpaceNet
Jun-2005	**OpenPKG 2.4**
Oct-2005	**OpenPKG 2.5**
Dec-2005	OpenPKG GmbH, OpenPKG Registry
Mar-2006	OpenPKG Websites 2.0
Jun-2006	**OpenPKG 2.6**
...	...

Some people have entirely too much **free time** on their hands.
— Gene Spafford

About Project: Engineering Phases

- There are three types of recurring and overlapping phases in OpenPKG:
  - Development (Dev)
  - Release Engineering (RE)
  - Security Engineering (SE)
- Development Phase: implement new features, major changes, work-off packaging, ...
- Release Engineering Phase: fix building of packages, prepare release documents, ...
- Security Engineering Phase: ongoing effort to track security issues, backport and prepare patches, write security advisories, ...



Recursive, adj.; see Recursive.

Engineering Phases: Release Engineering

- Release Engineering is the recurring procedure where a new OpenPKG release is made.
- The frequency of 4 months is a balance between...
  - making the latest vendor software versions available for production environments.
  - providing a stable and consistent set of packages.
  - able to support risk free security updates for existing installations.
  - allow reproducable installations through fixated package versions.
  - having a limited amount of sponsored and contributed manpower and resources available.
- The Release Engineering steps mainly involve:
  - updating the OpenPKG build farm to the latest OS vendor versions/patchlevels.
  - fixing all CORE/BASE/PLUS packages to work on all supported platforms.
  - blessing EVAL class packages for PLUS class if they work on all platforms in order to increase release extend.
  - rolling the source and binary package distribution on all platforms for CORE/BASE/PLUS.
  - quality testing the packages.
  - updating documentation and publically publishing the results.

A new release is where **old bad** assumptions are replaced by **new bad** assumptions.

Engineering Phases: Security Engineering

- Security Engineering is an important task in OpenPKG because every release has a life-time (usually 8 months).
- During the release life-time, existing installations are maintained with on-demand security updates.
- Deploying an OpenPKG security update is risk free, i.e., the user is guaranteed that no incompatible functional change or even new feature exists in any release update packages.
- The OpenPKG project achieves this by fully back-porting security fixes to the actually packaged vendor version. There is no simple vendor version upgrade made.
- The OpenPKG community is informed through public security advisories, summarizing the security issue and providing detailed information about affected releases and package versions.
- The OpenPKG project participates in closed vendor forums to get earliest possible notifications about security issues and to share own informations with other vendors.
- As a result of the ongoing OpenPKG security engineering process, the community gets security fixes as fast as possible.

The only **secure** computer is one that's unplugged, locked in a safe, and buried 20 feet under the ground in a secret location... and I'm **not** even too sure about that one.

— Dennis Huges, FBI.

Who's Who? (1)
Ralf S. Engelschall

- Person Details:
  - Name: Ralf S. Engelschall
  - Born: November 17th, 1972
  - Nationality: German
  - Status: married, 2 children
  - Profession: Computer Scientist
  - Experience: 18 years of computing

- Ralf S. Engelschall is the founder and principal architect of the OpenPKG project.
- He is the author of about 90% of all OpenPKG packages.
- Together with the OpenPKG Foundation he holds the copyright on OpenPKG.
- He is also founder and president of the OpenPKG Foundation e.V.

- His other well-known Open Source Software achievements:
  - founder, principal architect and author of OSSP.
  - co-founder and developer at OpenSSL.
  - founder and author of Apache mod_ssl, author of Apache mod_rewrite, mod_dso and APACI.
  - developer at FreeBSD.

A hacker does for **love** what others would **not** do for **money**.

Ralf S. Engelschall
rse@engelschall.com
rse.engelschall.com

Who's Who? (2)
OpenPKG Foundation e.V.

- The social community around OpenPKG forms up in the OpenPKG Foundation e.V. http://www.openpkg.net/
- Excerpt from the Foundation constitution: "Intention of the OpenPKG Foundation e.V. is the ideational, financial, material and manned support of the Open Software Project OpenPKG."
- The OpenPKG Foundation is a non-profit organisation, founded 2005 by Ralf S. Engelschall, Thomas Lotterer and OpenPKG developers.

When I was a boy I was told **anybody** can become **president**. I'm beginning to believe it… — *Clarence Darrow*

- The OpenPKG Foundation is established as an association under German law and regulated by a registered association constitution and companion bylaws following democratic rules.

Teamwork is essential:
There is always one you can **blame** it on.

Who's Who? (3) Sponsors

- In addition to the development efforts provided individuals during their free time, the OpenPKG project is backed by sponsors from the IT industry.
- The sponsors mainly provide:
  - human resources (man-power)
  - hardware resources (servers)
  - hosting resources (datacenter)
  - network resources (Internet)
- Between 1992 and 2000, the primary sponsor of OpenPKG's predecessors was **sd&m**.
  http://www.sdm.de/
- Between 2000 and 2005, the primary sponsor of OpenPKG was **Cable & Wireless**.
  http://www.cw.com/

- Since 2005, the primary sponsors of OpenPKG are:
  - **OpenPKG Foundation e.V.**
    http://www.openpkg.net/
    providing human resources and hardware resources.

  - **SpaceNet AG**
    http://www.space.net/
    providing hosting resources and network resources.

If a **trainstation** is where trains stop, what is a **workstation**?

Part IV: User Perspectives

OpenPKG RPM Crash-Course
OpenPKG Live (Demonstration)
Package Lifecycle

A **supercomputer** is a machine, that runs an **endless** loop in just 2 seconds.

OpenPKG RPM Crash-Course

- Bootstrapping Instance:
```
$ sh openpkg-*.src.sh
$ sh openpkg-*.*-*.sh
```
- Installing Packages:
```
$ openpkg rpm –rebuild \
  foo-*.src.rpm
# openpkg rpm -Uvh \
  foo-*.*-*.rpm
```
- Starting/Stopping Services:
```
# openpkg rc foo stop start
# openpkg rc foo status
```
- Removing Packages:
```
# openpkg rpm -e foo
```
- Removing Instance:
```
# openpkg rpm -e `openpkg rpm \
  -q --whatrequires openpkg`
# openpkg rpm -e openpkg
```
- Query Information:
```
$ openpkg rpm –qa
$ openpkg rpm -qi foo
$ openpkg rpm -qlv foo
$ openpkg rpm –qf \
  /path/to/file
$ openpkg rpm -qpi \
  foo-*.rpm
$ openpkg rpm –qp \
  --requires foo-*.rpm
```
- Verify Integrity:
```
# openpkg rpm -V foo
# openpkg rpm –Va
```
- Reading RPM Manual:
```
$ openpkg man rpm
```

Everybody **falls** the first time.
It doesn't mean anything.
— The Matrix

OpenPKG Live (1)

- Build binary from source bootstrap package

```
$ TMPDIR=/var/tmp; export TMPDIR; cd $TMPDIR
$ ftp ftp.openpkg.org
Connected to ftp.openpkg.org.
220 ftp.openpkg.org OpenPKG Anonymous FTP Server ready.
Name (ftp.openpkg.org): anonymous
331 Anonymous login ok, send your email address as password.
Password: you@example.com
230- [...] Welcome to OpenPKG.org! [...]
230 Anonymous access granted, restrictions apply.
ftp> bin
200 Type set to I.
ftp> cd release/2.5/SRC
ftp> get openpkg-2.5.0-2.5.0.src.sh
ftp> bye
221 Goodbye.
$ sh ./openpkg-2.5.0-2.5.0.src.sh --tag=opkg \
  --prefix=/usr/opkg --user=opkg --group=opkg
OpenPKG 2.5-RELEASE Source Bootstrap Package, version 2.5.0
Building for prefix /usr/opkg on current platform
++ extracting OpenPKG source distribution
++ building OpenPKG binary distribution
[...]
$ ls -l openpkg-*
-rw-r--r--  1 foo foo 18558976 Oct 20 10:20 openpkg-2.5.0-2.5.0.src.sh
-rw-r--r--  1 foo foo 16997568 Oct 20 10:20 openpkg-2.5.0-2.5.0.src.rpm
-rw-r--r--  1 foo foo  6230016 Oct 20 10:20 openpkg-2.5.0-2.5.0.ix86-freebsd5.4-opkg.sh
-rw-r--r--  1 foo foo  5989118 Oct 20 10:20 openpkg-2.5.0-2.5.0.ix86-freebsd5.4-opkg.rpm
$ _
```

"The idea is to **fall** and **miss** the ground."
— Douglas Adams, A Hitchhiker's Guide to the galaxy.

OpenPKG Live (2)

- Install binary bootstrap package to create instance

```
$ su –
Password: *****
# sh ./openpkg-2.5.0-2.5.0.ix86-freebsd5.4-opkg.sh
OpenPKG 2.5-RELEASE Binary Bootstrap Package, version 2.5.0
Built for prefix /tmp/openpkg on target platform ix86-freebsd5.4
++ hooking OpenPKG instance into system environment
++ creating OpenPKG instance root directory "/usr/opkg"
[...]
# exit
$ ls –l /usr/opkg
-rw-r--r--   1 opkg  opkg    911 Oct 20 10:20 README
drwxr-xr-x   6 opkg  opkg    512 Oct 20 10:20 RPM
drwxr-xr-x   2 opkg  opkg    512 Oct 20 10:20 bin
drwxr-xr-x   2 opkg  opkg    512 Oct 20 10:20 cgi
drwxr-xr-x   4 opkg  opkg    512 Oct 20 10:20 etc
drwxr-xr-x   3 opkg  opkg    512 Oct 20 10:20 include
drwxr-xr-x   2 opkg  opkg    512 Oct 20 10:20 info
drwxr-xr-x   3 opkg  opkg    512 Oct 20 10:20 lib
drwxr-xr-x   3 opkg  opkg    512 Oct 20 10:20 libexec
drwxr-xr-x  10 opkg  opkg    512 Oct 20 10:20 local
drwxr-xr-x  20 opkg  opkg    512 Oct 20 10:20 man
drwxr-xr-x   2 opkg  opkg    512 Oct 20 10:20 pub
drwxr-xr-x   2 opkg  opkg    512 Oct 20 10:20 sbin
drwxr-xr-x   2 opkg  opkg    512 Oct 20 10:20 share
drwxr-xr-x   2 opkg  opkg    512 Oct 20 10:20 var
$ /usr/opkg/bin/openpkg rpm –qa
openpkg-2.5.0-2.5.0
gpg-pubkey-63c4cb9f-3c591eda
$ _
```

A computer scientist is someone who **fixes** things that **aren't** broken.

OpenPKG Live (3)

- Install OpenPKG package for GNU Bash (example)

```
$ /usr/opkg/bin/openpkg rpm --rebuild \
  ftp://ftp.openpkg.org/release/2.5/SRC/bash-3.0.16-2.5.0.src.rpm
Installing ftp://ftp.openpkg.org/release/2.5/SRC/bash-3.0.16-2.5.0.src.rpm
[...]
Wrote: /usr/opkg/RPM/PKG/bash-3.0.16-2.5.0.ix86-freebsd5.4-opkg.rpm
$ su -
# /usr/opkg/bin/openpkg rpm -Uvh \
  /usr/opkg/RPM/PKG/bash-3.0.16-2.5.0.ix86-freebsd5.4-opkg.rpm
Preparing...                ########################################### [100%]
   1:bash                   ########################################### [100%]
# exit
$ /usr/opkg/bin/openpkg rpm -qlv bash
-rwxr-xr-x 1 opkg opkg    539068 Oct 20 10:20 /usr/opkg/bin/bash
drwxr-xr-x 2 opkg opkg         0 Oct 20 10:20 /usr/opkg/etc/bash
-rw-r--r-- 1 opkg opkg      2756 Oct 20 10:20 /usr/opkg/etc/bash/profile
-rw-r--r-- 1 opkg opkg    342251 Oct 20 10:20 /usr/opkg/info/bash.info
-rw-r--r-- 1 opkg opkg    228383 Oct 20 10:20 /usr/opkg/man/man1/bash.1
$ /usr/opkg/bin/openpkg rpm -qi bash
Name:     bash                      Source RPM:   bash-3.0.16-2.5.0.src.rpm
Version:  3.0.16                    Signature:    md5:e943b1ae7004def2baa91563341ad9d3
Release:  2.5.0                     Build Host:   foo.example.com
Group:    Shell                     Build System: ix86-freebsd5.4
Class:    CORE                      Build Time:   Wed Oct 20 10:20:00 2005
Distrib:  OpenPKG                   Install Time: Wed Oct 20 10:20:30 2005
License:  GPL                       Install Size: 1112458 bytes
Packager: The OpenPKG Project       Relocations:  /usr/opkg
Vendor:   Free Software Foundation
[...]
$ _
```

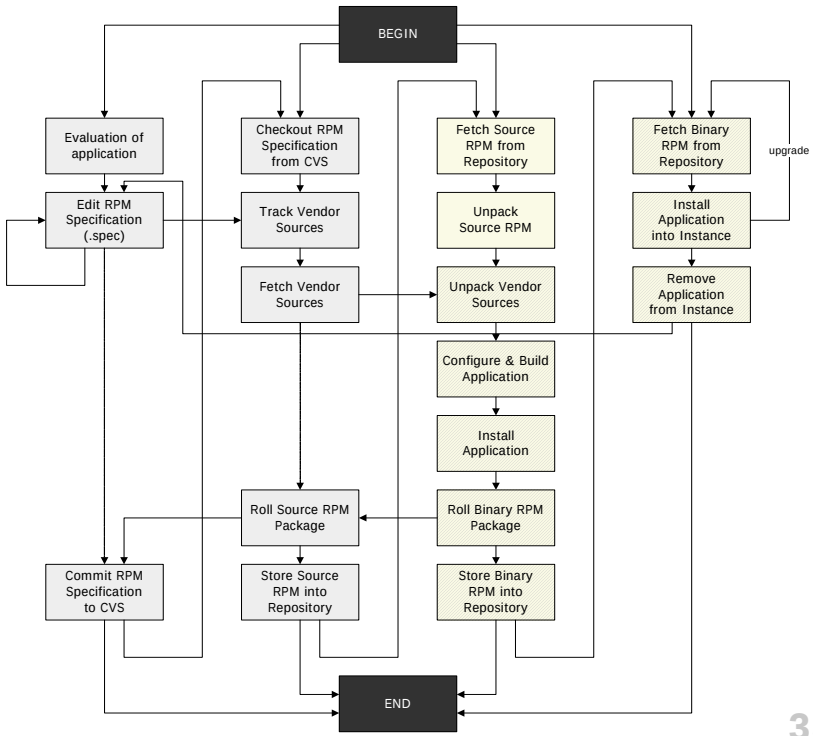
Seek **simplicity** but distrust it.
— A. N. Whitehead

Package Lifecycle (1)

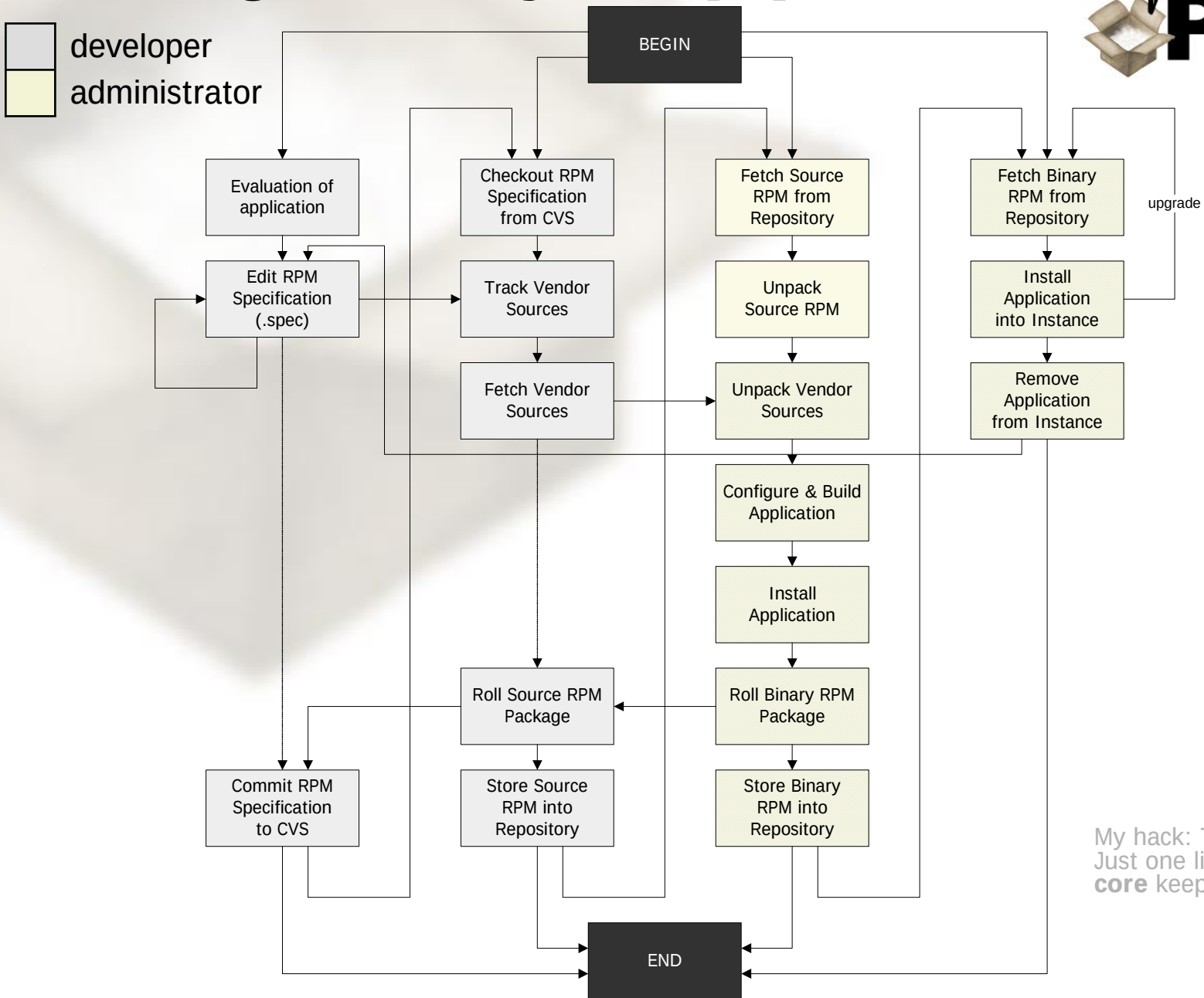
- The lifecycle of a package is the most important part to understand in OpenPKG.
- In OpenPKG, the lifecycle is an extended RPM package lifecycle because of extensions to RPM.
- The lifecycle consists of overlapping steps performed by two parties:
  - OpenPKG developers creating packages.
  - OpenPKG administrators deploying packages.

You can **check out** any time you like, but you can never **leave**.
— The Eagles, Hotel California

- The developer performs most of the administrator steps during build-time and run-time testing.
- The administrator repeats some of the developer steps during building from source.



Package Lifecycle (2)



My hack: This **universe**.
Just one little problem:
core keeps dumping.

Part V: Developer Perspectives

Package Components
Package Specification
Package Building
Development: Version Tracking
Development: CVS Repository
Development: Build Farm

Computer science is no more about **computers** than astronomy is about **telescopes**.
— *E. W. Dijkstra*

Package Components

- Package Specification:
  - central OpenPKG RPM packaging information (*name*`.spec`)
- Vendor Sources:
  - vendor tarball (*name*`-`*version*`.tar.gz`)
  - vendor patches (*name*`-`*version*`.patch`)
- Extra Packaging Files
  - packager or third-party patches (*name*`.patch[.`*tag*`]`)
  - run-command scripts, FSL configurations, etc. (`rc.`*name*, `fsl.`*name*)
  - default configuration files (*name*`.conf`, ...)
  - ...

> **"UNIX** is **simple**.
> It just takes a genius to understand its simplicity."
> — *Dennis Ritchie*

```
-rw-r--r-- 1 rse   openpkg     6162 Mar 27 09:14 bash.spec
-rw-r--r-- 1 rse   openpkg     5305 Jan 23 13:47 bash.patch
-rw-r--r-- 1 rse   openpkg     2752 Feb 18 11:30 profile
-rw-r--r-- 1 rse   openpkg  1956216 Feb 24 23:02 bash-3.0.tar.gz
-rw-r--r-- 1 rse   openpkg     1132 Feb 24 23:02 bash30-001
-rw-r--r-- 1 rse   openpkg      755 Feb 24 23:02 bash30-002
-rw-r--r-- 1 rse   openpkg     2356 Feb 24 23:02 bash30-003
-rw-r--r-- 1 rse   openpkg     1110 Feb 24 23:02 bash30-004
-rw-r--r-- 1 rse   openpkg     2217 Feb 24 23:02 bash30-005
-rw-r--r-- 1 rse   openpkg     3155 Feb 24 23:02 bash30-006
-rw-r--r-- 1 rse   openpkg     1072 Feb 24 23:02 bash30-007
  :           :        :        :    :        :        :
```

Package Specification (1)

- Every OpenPKG RPM package specification follows exactly the same structure and is strictly checked syntactically.
- The section ordering is:
  - macro defines
  - package headers
  - package options
  - source references
  - package dependencies
  - package description
  - version tracking
  - build preparation
  - configuration & building
  - installation
  - file determination
  - cleanup
  - deploy-time scripting

```
%build
    #   configure package
    ( #   force disabled wide-character support
      echo "ac_cv_header_wchar_h=no"
      echo "ac_cv_header_wctype_h=no"
      echo "ac_cv_
      #   force di
      echo "ac_cv_
      echo "ac_cv_
      echo "ac_cv_
      echo "ac_cv_
      echo "ac_cv_
    ) >config.cach
    CC="%{l_cc}" \
    CFLAGS="%{l_cf
    ./configure \
        --cache-fi
        --prefix=%
        --without
        --without
    %{l_shtool} su
        -e 's;^\((
        pathnames
    %{l_shtool} su
        -e 's;/etc
        doc/bash.1

    #   build pack
    %{l_make} %{l

%install
    #   install pa
    rm -rf $RPM_BU
    %{l_make} %{l
        prefix=$RP

    #   strip down
    rm -f $RPM_BU
    rm -f $RPM_BU
    rm -f $RPM_BU
    strip $RPM_BU

    #   install g
    %{l_shtool} mk
        $RPM_BUILD
    %{l_shtool} in
        %{SOURCE

    #   determine
    %{l_rpmtool} f
        %{l_files_
        '%config %

%files -f files

%clean
    rm -rf $RPM_BU

%post
    if [ ".$1" =
        #   displa
        if [ -f /e
            if [ "
                (
                )
            fi
        fi
    fi
```

```
    #   package version
    %define       V_base_real 3.0
    %define       V_base_comp 30
    %define       V_plvl_raw  13
    %define       V_plvl_pad  013

    #   package information
    Name:         bash
    Summary:      Bourne-Again Shell
    URL:          http://cnswww.cns.cwru.edu/~chet/bash/bashtop.html
    Vendor:       Free Software Foundation
    Packager:     The OpenPKG Project
    Distribution: OpenPKG
    Class:        CORE
    Group:        Shell
    License:      GPL
    Version:      %{V_base_real}.%{V_plvl_raw}
    Release:      2.2.0

    #   list of sources
    Source0:      ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}.tar.gz
    Source1:      profile
    Patch0:       bash.patch
    Patch1:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-001
    Patch2:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-002
    Patch3:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-003
    Patch4:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-004
    Patch5:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-005
    Patch6:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-006
    Patch7:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-007
    Patch8:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-008
    Patch9:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-009
    Patch10:      ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-0107
    Patch11:      ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-0117
    Patch12:      ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-012
    Patch13:      ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-013

    #   build information
    Prefix:       %{l_prefix}
    BuildRoot:    %{l_buildroot}
    BuildPreReq:  OpenPKG, openpkg >= 20040130
    PreReq:       OpenPKG, openpkg >= 20040130
    AutoReq:      no
    AutoReqProv:  no

    %description
        Bash (Bourne-Again Shell) is an sh-compatible command language interpreter
        that executes commands read from the standard input or from a file. Bash
        also incorporates useful features from the Korn and C shells (ksh and csh).
        Bash is intended to be a conformant implementation of the IEEE POSIX Shell
        and Tools specification (IEEE Working Group 1003.2).

    %track
        prog bash = {
            version   = %{V_base_real}
            url       = ftp://ftp.cwru.edu/pub/bash/
            regex     = bash-(__VER__)\.tar\.gz
        }
        prog bash:patches = {
            version   = %{V_base_comp}-%{V_plvl_pad}
            url       = ftp://ftp.cwru.edu/pub/bash/
            regex     = (bash-\d+\.\d+[a-z]+-patches)
            url       = ftp://ftp.cwru.edu/pub/bash/__NEWVER__/
            regex     = bash(\S+-\d+)
        }

    %prep
        %setup -q -n bash-%{V_base_real}
        %patch -p0 -P 0 1 2 3 4 5 6 7 8 9 10 11 12 13
        %{l_shtool} subst \
            -e 's;@l_openpkg_release@;%{l_openpkg_release};' \
            version.c
```

Politics is for the moment, an **equation** lasts **eternity**.
— *Albert Einstein*

Package Specification (2)

- In detail: Defines, Headers, Sources, Dependencies

```
#   package version
%define       V_base_real 3.0
%define       V_base_comp 30
%define       V_plvl_raw  16
%define       V_plvl_pad  016

#   package information
Name:         bash
Summary:      Bourne-Again Shell
URL:          http://cnswww.cns.cwru.edu/~chet/bash/bashtop.html
Vendor:       Free Software Foundation
Packager:     The OpenPKG Project
Distribution: OpenPKG
Class:        CORE
Group:        Shell
License:      GPL
Version:      %{V_base_real}.%{V_plvl_raw}
Release:      2.5.0

#   list of sources
Source0:      ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}.tar.gz
Source1:      profile
Patch0:       bash.patch
Patch1:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-001
Patch2:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-002
Patch3:       ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-003
...           ...
Patch15:      ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-015
Patch16:      ftp://ftp.cwru.edu/pub/bash/bash-%{V_base_real}-patches/bash%{V_base_comp}-016

#   build information
Prefix:       %{l_prefix}
BuildRoot:    %{l_buildroot}
BuildPreReq:  OpenPKG, openpkg >= 2.5.0
PreReq:       OpenPKG, openpkg >= 2.5.0
AutoReq:      no
```

What you see
is **all you get**.
— Brian Kernighan

Package Specification (3)

- In detail: Description, Tracking, Preparation

```
%description
    Bash (Bourne-Again Shell) is an sh-compatible command language interpreter
    that executes commands read from the standard input or from a file. Bash
    also incorporates useful features from the Korn and C shells (ksh and csh).
    Bash is intended to be a conformant implementation of the IEEE POSIX Shell
    and Tools specification (IEEE Working Group 1003.2).

%track
    prog bash = {
        version   = %{V_base_real}
        url       = ftp://ftp.cwru.edu/pub/bash/
        regex     = bash-(__VER__)\.tar\.gz
    }
    prog bash:patches = {
        version   = %{V_base_comp}-%{V_plvl_pad}
        url       = ftp://ftp.cwru.edu/pub/bash/
        regex     = (bash-\d+\.\d+[a-z]+-patches)
        url       = ftp://ftp.cwru.edu/pub/bash/__NEWVER__/
        regex     = bash(\S+-\d+)
    }

%prep
    #   unpack and patch distribution
    %setup -q -n bash-%{V_base_real}
    %patch -p0 -P 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

    #   brand with OpenPKG release and fix patchlevel
    %{l_shtool} subst \
        -e 's;@l_openpkg_release@;%{l_openpkg_release};' \
        version.c
    %{l_shtool} subst \
        -e 's;\(PATCHLEVEL\) 0;\1 %{V_plvl_raw};' \
        patchlevel.h
```

Beware of bugs in the above code; I have only **proved** it correct, not tried it.
— *D.E. Knuth*

Package Specification (4)

- In detail: Configuration and Building

```
%build
    #   configure package
    ( #   force disabled wide-character support
      echo "ac_cv_header_wchar_h=no"
      echo "ac_cv_header_wctype_h=no"
      echo "ac_cv_func_mbsrtowcs=no"
      #   force disabled internationalization support
      echo "ac_cv_header_libintl_h=no"
      echo "ac_cv_func_gettext=no"
      echo "ac_cv_func_textdomain=no"
      echo "ac_cv_func_bindtextdomain=no"
      echo "ac_cv_lib_intl_bindtextdomain=no"
    ) >config.cache
    CC="%{l_cc}" \
    CFLAGS="%{l_cflags -O}" \
    ./configure \
        --cache-file=./config.cache \
        --prefix=%{l_prefix} \
        --disable-multibyte \
        --enable-debugger \
        --without-gnu-malloc \
        --without-curses \
        --disable-nls
    %{l_shtool} subst \
        -e 's;^\(#define.*SYS_PROFILE["^]*\).*;\1 "%{l_prefix}/etc/bash/profile";' \
        pathnames.h
    %{l_shtool} subst \
        -e 's;/etc/profile;%{l_prefix}/etc/bash/profile;' \
        doc/bash.1

    #   build package
    %{l_make} %{l_mflags}
```

Try to **understand** everything,
but **believe** nothing!

Package Specification (5)

- In detail: Installation, File Determination, Cleanup

```
%install
    #   install package
    rm -rf $RPM_BUILD_ROOT
    %{l_make} %{l_mflags} install \
        prefix=$RPM_BUILD_ROOT%{l_prefix}

    #   strip down installation
    rm -f $RPM_BUILD_ROOT%{l_prefix}/info/dir
    rm -f $RPM_BUILD_ROOT%{l_prefix}/man/man1/bashbug.1
    rm -f $RPM_BUILD_ROOT%{l_prefix}/bin/bashbug
    strip $RPM_BUILD_ROOT%{l_prefix}/bin/bash

    #   install global configuration
    %{l_shtool} mkdir -f -p -m 755 \
        $RPM_BUILD_ROOT%{l_prefix}/etc/bash
    %{l_shtool} install -c -m 644 %{l_value -s -a} \
        %{SOURCE profile} $RPM_BUILD_ROOT%{l_prefix}/etc/bash/

    #   determine installation files
    %{l_rpmtool} files -v -ofiles -r$RPM_BUILD_ROOT \
        %{l_files_std} \
        '%config %{l_prefix}/etc/bash/profile'

%files -f files

%clean
    rm -rf $RPM_BUILD_ROOT
```

The number of UNIX installations
has grown to **10**, with **more expected**.
— *The UNIX Programmer's Manual,*
2nd Edition, June, 1972

Package Specification (6)

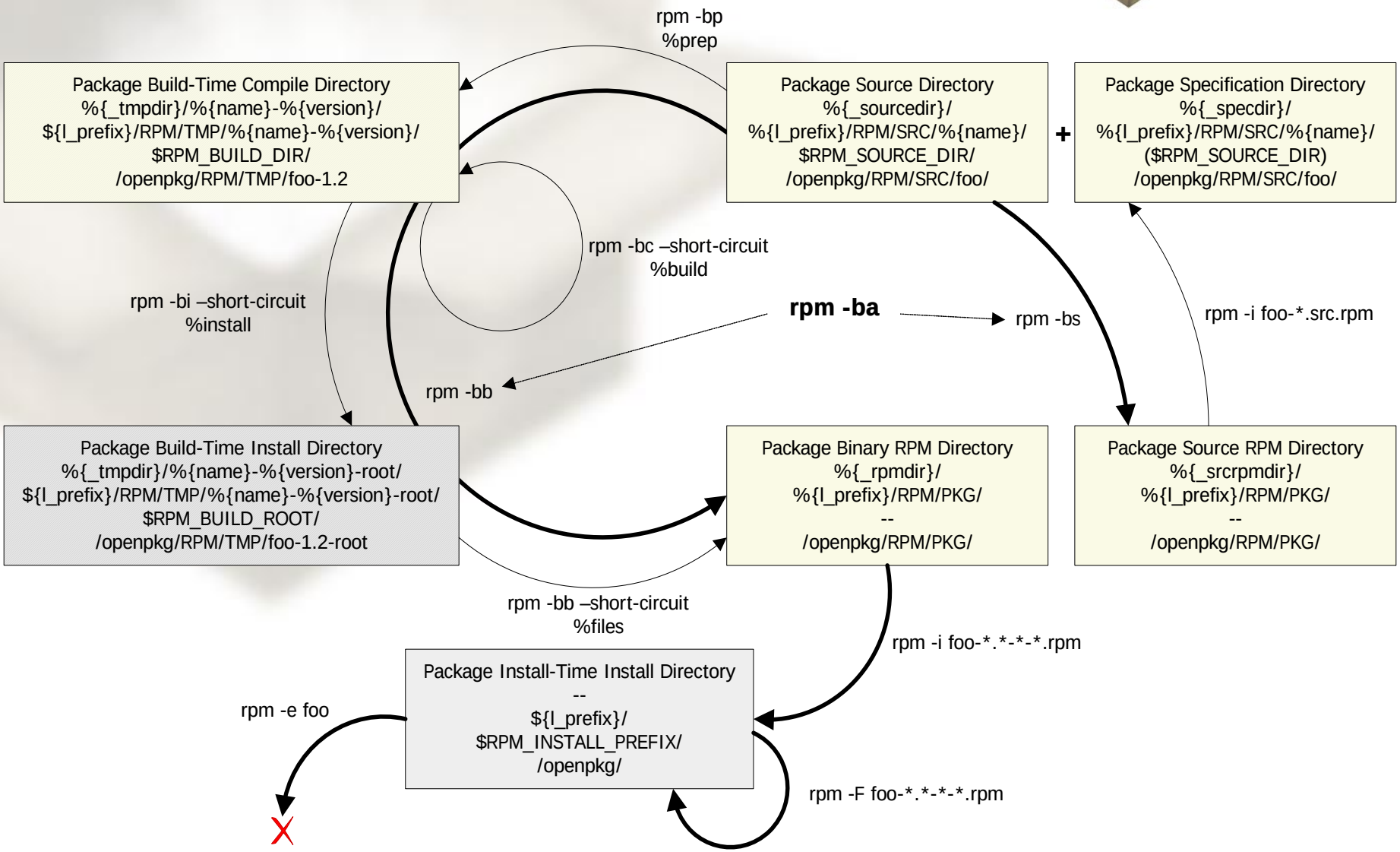
- In detail: Post-Installation Processing

```
%post
    if [ ".$1" = .1 ]; then
        #   display note about login shell prerequisite
        if [ -f /etc/shells ]; then
            if [ ".`grep $RPM_INSTALL_PREFIX/bin/bash /etc/shells`" = . ]; then
                ( echo "Hint: To use $RPM_INSTALL_PREFIX/bin/bash as the login"
                  echo "shell for users, please add this path to /etc/shells."
                ) | %{l_rpmtool} msg -b -t notice
            fi
        fi
    fi
```

I conclude that there are two ways of constructing a software design: One way is to make it so **simple** that there are **obviously no deficiencies** and the other way is to make it so **complicated** that there are **no obvious deficiencies**.

— C.A.R.Hoare

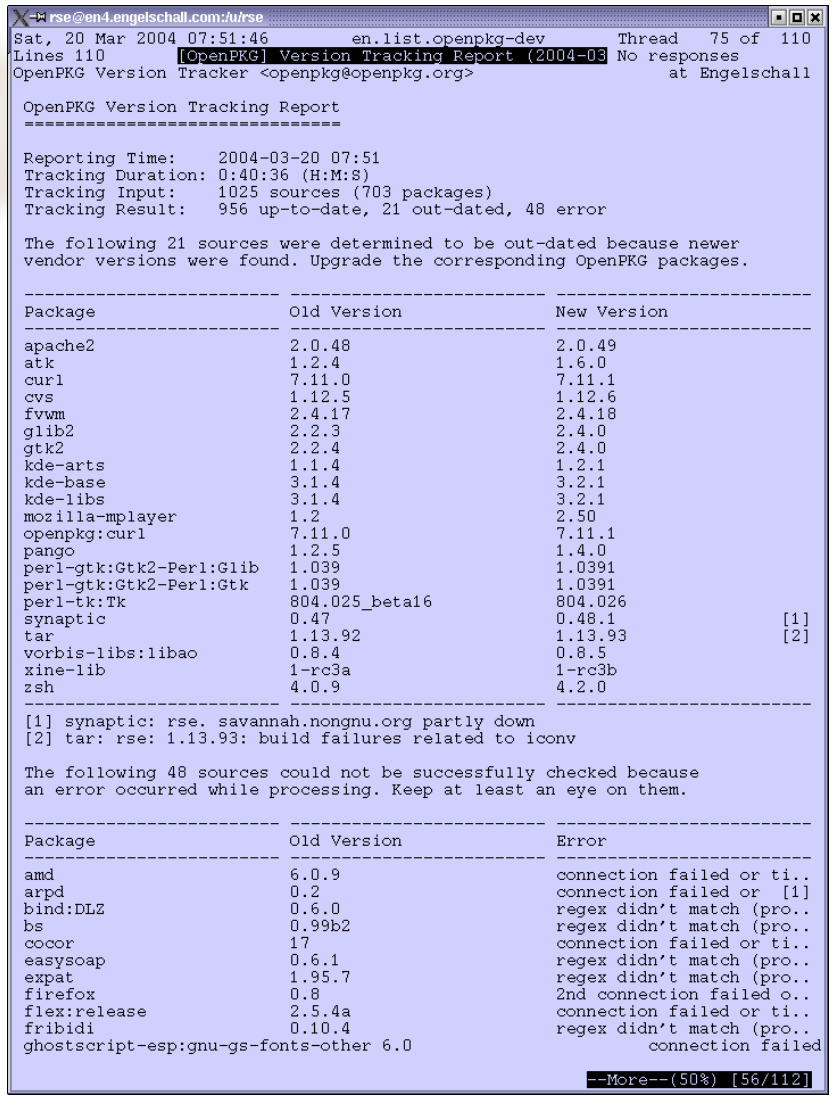
Package Building: RPM Control/Data Flow



Development: Version Tracking

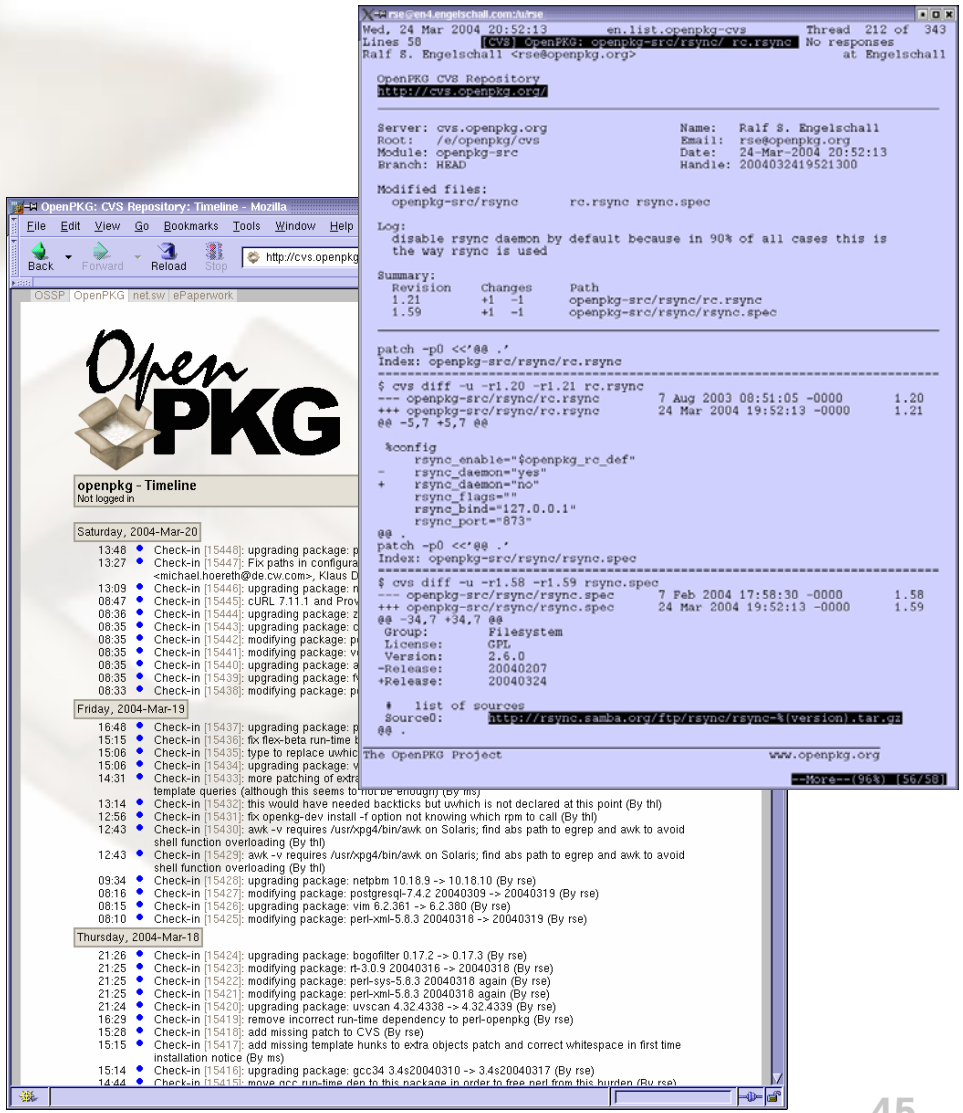
Life is just a **beta-version**.
Don't expect it to be **bug-free**.

- OpenPKG RPM supports a custom section %track which contains vcheck(1) configurations.
- A vcheck(1) configuration is:
  - last known version
  - URL where the versions are referenced
  - regular expression how the versions can be extracted from the text under the URL
- All 800 OpenPKG packages contain a %track section for checking all external source files of a package.
- On a bi-daily basis all %track sections are executed and a report sent to the OpenPKG developers.
- See also: openpkg-dev@openpkg.org

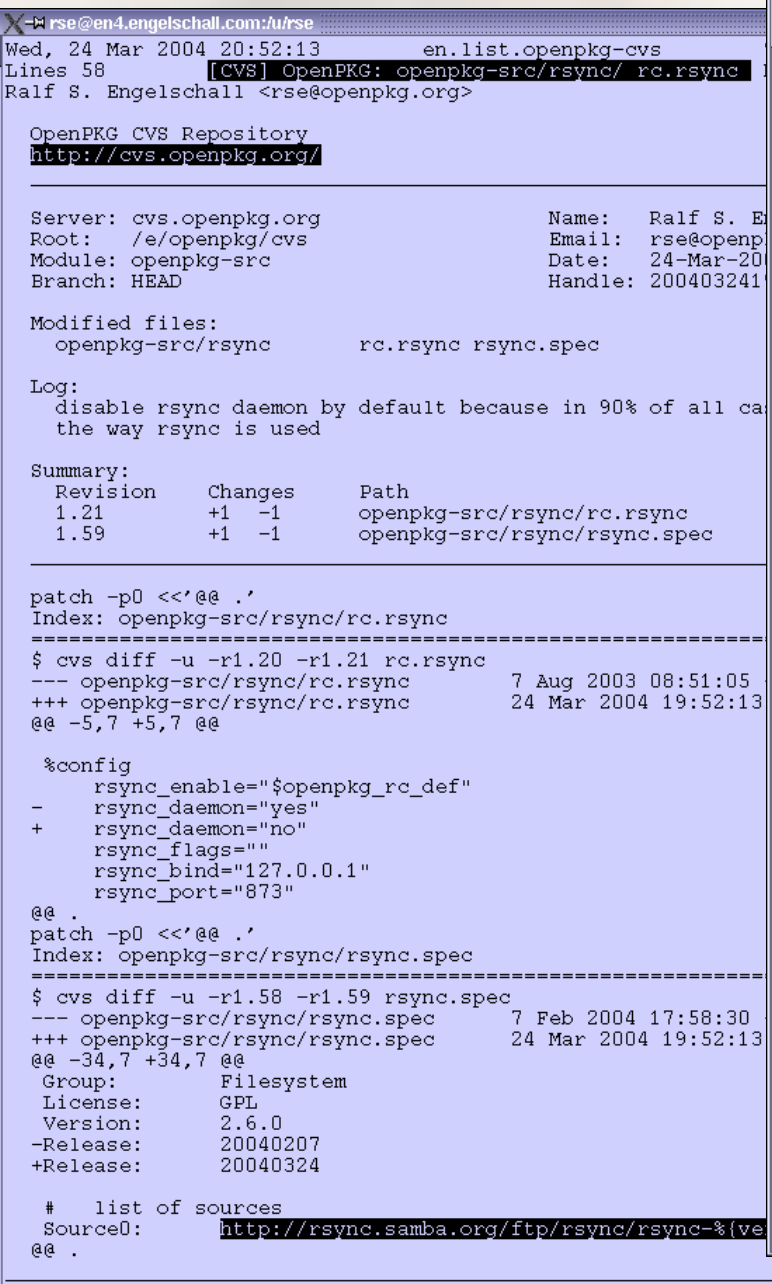


Development: CVS Repository

- All sources of OpenPKG are stored in a central CVS based repository system.
- Every “Commit” to the repository is real-time tracked both with detailed reports via Email and on-line via CVSTrac.
- Every OpenPKG release is an own “branch” in the repository.
- See also: http://cvs.openpkg.org/ openpkg-cvs@openpkg.org



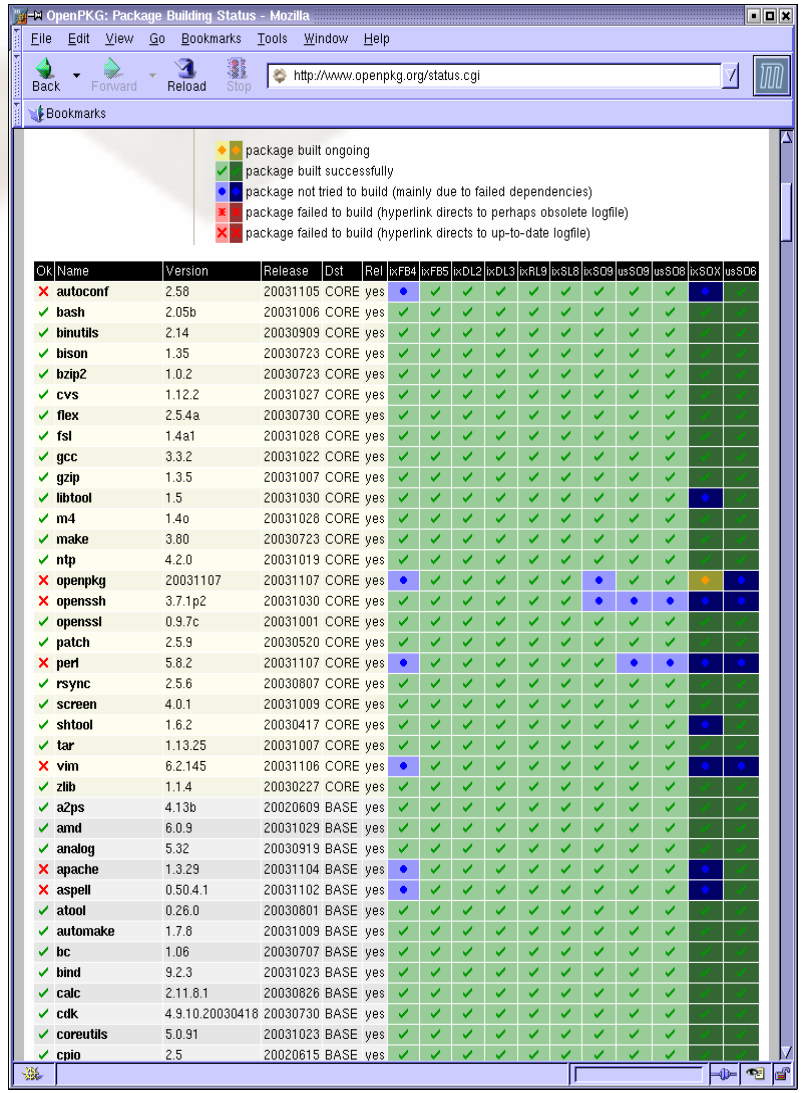
Murphy's Law is recursive: Washing your car to make it rain doesn't work.


rse@en4.engelschall.com:/u/rse
Wed, 24 Mar 2004 20:52:13 en.list.openpkg-cvs
Lines 58 [CVS] OpenPKG: openpkg-src/rsync/ rc.rsync
Ralf S. Engelschall <rse@openpkg.org>
OpenPKG CVS Repository
http://cvs.openpkg.org/
Server: cvs.openpkg.org Name: Ralf S. E
Root: /e/openpkg/cvs Email: rse@openp
Module: openpkg-src Date: 24-Mar-20
Branch: HEAD Handle: 200403241
Modified files:
openpkg-src/rsync rc.rsync rsync.spec
Log:
disable rsync daemon by default because in 90% of all ca
the way rsync is used
Summary:
Revision Changes Path
1.21 +1 -1 openpkg-src/rsync/rc.rsync
1.59 +1 -1 openpkg-src/rsync/rsync.spec
patch -p0 <<'@@ .'
Index: openpkg-src/rsync/rc.rsync
$ cvs diff -u -r1.20 -r1.21 rc.rsync
--- openpkg-src/rsync/rc.rsync 7 Aug 2003 08:51:05
+++ openpkg-src/rsync/rc.rsync 24 Mar 2004 19:52:13
@@ -5,7 +5,7 @@
%config
rsync_enable="$openpkg_rc_def"
- rsync_daemon="yes"
+ rsync_daemon="no"
rsync_flags=""
rsync_bind="127.0.0.1"
rsync_port="873"
@@ .
patch -p0 <<'@@ .'
Index: openpkg-src/rsync/rsync.spec
$ cvs diff -u -r1.58 -r1.59 rsync.spec
--- openpkg-src/rsync/rsync.spec 7 Feb 2004 17:58:30
+++ openpkg-src/rsync/rsync.spec 24 Mar 2004 19:52:13
@@ -34,7 +34,7 @@
Group: Filesystem
License: GPL
Version: 2.6.0
-Release: 20040207
+Release: 20040324
list of sources
Source0: http://rsync.samba.org/ftp/rsync/rsync-%{ve
@@ .
The OpenPKG Project www.openpkg.org
--More--(96%) [56/58]


OpenPKG: CVS Repository: Timeline - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://cvs.openpkg.org/timeline?d=30&e=2004-Mar-20&c=2&px=&s=1&dt=1&m=1&x=1
OSSP OpenPKG net.sw ePaperwork
OpenPKG
CABLE & WIRELESS
Internet Services
openpkg - Timeline
Not logged in
[Home] [Search]
Saturday, 2004-Mar-20
13:48 Check-in [15448]: upgrading package: php5 5.0.0b4 -> 5.0.0RC1 (By rse)
13:27 Check-in [15447]: Fix paths in configuration and CGI script. Submitted by: Michael Hoereth <michael.hoereth@de.cw.com>, Klaus Doll <klaus.doll@de.cw.com> (By rse)
13:09 Check-in [15446]: upgrading package: nail 10.6 -> 10.7 (By rse)
08:47 Check-in [15445]: cURL 7.11.1 and Provides bump (By rse)
08:36 Check-in [15444]: upgrading package: zsh 4.0.9 -> 4.2.0 (By rse)
08:35 Check-in [15443]: upgrading package: curl 7.11.0 -> 7.11.1 (By rse)
08:35 Check-in [15442]: modifying package: perl-gtk-5.8.3 20040318 -> 20040320 (By rse)
08:35 Check-in [15441]: modifying package: vorbis-libs-1.0.1 20040207 -> 20040320 (By rse)
08:35 Check-in [15440]: upgrading package: apache2 2.0.48 -> 2.0.49 (By rse)
08:35 Check-in [15439]: upgrading package: fvwm 2.4.17 -> 2.4.18 (By rse)
08:33 Check-in [15438]: modifying package: perl-tk-5.8.3 20040318 -> 20040320 (By rse)
Friday, 2004-Mar-19
16:48 Check-in [15437]: upgrading package: petidomo 4.0b5 -> 4.0b6 (By thl)
15:15 Check-in [15436]: fix flex-beta run-time broken on solaris (By thl)
15:06 Check-in [15435]: type to replace uwhich bashification (By thl)
15:06 Check-in [15434]: upgrading package: vim 6.2.380 -> 6.2.382 (By thl)
14:31 Check-in [15433]: more patching of extra object files, attempting to close the gaps leading to missing object template queries (although this seems to not be enough) (By ms)
13:14 Check-in [15432]: this would have needed backticks but uwhich is not declared at this point (By thl)
12:56 Check-in [15431]: fix openkg-dev install -f option not knowing which rpm to call (By thl)
12:43 Check-in [15430]: awk -v requires /usr/xpg4/bin/awk on Solaris; find abs path to egrep and awk to avoid shell function overloading (By thl)
12:43 Check-in [15429]: awk -v requires /usr/xpg4/bin/awk on Solaris; find abs path to egrep and awk to avoid shell function overloading (By thl)
09:34 Check-in [15428]: upgrading package: netpbm 10.18.9 -> 10.18.10 (By rse)
08:16 Check-in [15427]: modifying package: postgresql-7.4.2 20040309 -> 20040319 (By rse)
08:15 Check-in [15426]: upgrading package: vim 6.2.361 -> 6.2.380 (By rse)
08:10 Check-in [15425]: modifying package: perl-xml-5.8.3 20040318 -> 20040319 (By rse)
Thursday, 2004-Mar-18
21:26 Check-in [15424]: upgrading package: bogofilter 0.17.2 -> 0.17.3 (By rse)
21:25 Check-in [15423]: modifying package: rt-3.0.9 20040316 -> 20040318 (By rse)
21:25 Check-in [15422]: modifying package: perl-sys-5.8.3 20040318 again (By rse)
21:25 Check-in [15421]: modifying package: perl-xml-5.8.3 20040318 again (By rse)
21:24 Check-in [15420]: upgrading package: uvscan 4.32.4338 -> 4.32.4339 (By rse)
16:29 Check-in [15419]: remove incorrect run-time dependency to perl-openpkg (By rse)
15:28 Check-in [15418]: add missing patch to CVS (By rse)
15:15 Check-in [15417]: add missing template hunks to extra objects patch and correct whitespace in first time installation notice (By ms)
15:14 Check-in [15416]: upgrading package: gcc34 3.4s20040310 -> 3.4s20040317 (By rse)

Development: Build Farm

- OpenPKG packages are constantly tested on a large set of different platforms.
- For this a “build farm” is used (provided by the OpenPKG Foundation e.V.), consisting of machines which constantly fetch the latest OpenPKG-CURRENT and try to build changed packages.
- The result is a status page on the website which shows the latest status of each package on each platform.
- The developers watch this status page to see where something has to be fixed.
- See also: http://www.openpkg.org/status.cgi



Ok	Name	Version	Release	Dst	Rel
✗	autoconf	2.58	20031105	CORE	yes
✓	bash	2.05b	20031006	CORE	yes
✓	binutils	2.14	20030909	CORE	yes
✓	bison	1.35	20030723	CORE	yes
✓	bzip2	1.0.2	20030723	CORE	yes
✓	cvs	1.12.2	20031027	CORE	yes
✓	flex	2.5.4a	20030730	CORE	yes
✓	fsl	1.4a1	20031028	CORE	yes
✓	gcc	3.3.2	20031022	CORE	yes
✓	gzip	1.3.5	20031007	CORE	yes
✓	libtool	1.5	20031030	CORE	yes
✓	m4	1.4o	20031028	CORE	yes
✓	make	3.80	20030723	CORE	yes
✓	ntp	4.2.0	20031019	CORE	yes
✗	openpkg	20031107	20031107	CORE	yes
✗	openssh	3.7.1p2	20031030	CORE	yes
✓	openssl	0.9.7c	20031001	CORE	yes
✓	patch	2.5.9	20030520	CORE	yes
✗	perl	5.8.2	20031107	CORE	yes
✓	rsync	2.5.6	20030807	CORE	yes
✓	screen	4.0.1	20031009	CORE	yes
✓	shtool	1.6.2	20030417	CORE	yes
✓	tar	1.13.25	20031007	CORE	yes
✗	vim	6.2.145	20031106	CORE	yes
✓	zlib	1.1.4	20030227	CORE	yes
✓	a2ps	4.13b	20020609	BASE	yes
✓	amd	6.0.9	20031029	BASE	yes
✓	analog	5.32	20030919	BASE	yes
✗	apache	1.3.29	20031104	BASE	yes
✗	aspell	0.50.4.1	20031102	BASE	yes
✓	atool	0.26.0	20030801	BASE	yes
✓	automake	1.7.8	20031009	BASE	yes
✓	bc	1.06	20030707	BASE	yes
✓	bind	9.2.3	20031023	BASE	yes
✓	calc	2.11.8.1	20030826	BASE	yes
✓	cdk	4.9.10.20030418	20030730	BASE	yes
✓	coreutils	5.0.91	20031023	BASE	yes
✓	cpio	2.5	20020615	BASE	yes

Part VI: Some Gory Details

The “Bootstrap” (Package)
Run-Command Facility (RC)
OSSP fsl (Faking Syslog Library)

A **diplomat** is someone who can tell you to go to **hell** in such a way that you will **look forward** to the **trip**.

The “Bootstrap” (Package)

- OpenPKG technically consists of the essential “openpkg” RPM package plus 880 other RPM packages based on it.
- The “openpkg” package is called “the bootstrap” because it is
  - both a regular RPM package containing the RPM framework
  - and a elaborate bootstrapping procedure able to install itself with itself from scratch.
- This way OpenPKG RPM is 100% packaged by itself and especially is able to upgrade its RPM framework with itself.

All the **good things**
you want to do in your **life**
have to be **started** in the next
few hours, days or weeks.
— *Tom DeMarco*

- The bootstrapping works by...
  - emulating a minimum functional subset of RPM with a shell script.
  - building and installing the “openpkg” package content with the RPM emulation into a temporary area.
  - faking the rebuild and in-place re-installation of the “openpkg” package with the RPM from the temporary area in order to record RPM into its own RPM database.
  - rolling a bootstrapping binary shell script and binary RPM package from the temporary area.

Run-Command Facility (1) Overview

- OpenPKG provides a flexible and integrated Run-Command (RC) facility.
- The OpenPKG RC facility is...
  - based on ideas from the NetBSD 1.6 and FreeBSD 5 RC facility (no run-levels, `rc.d/` directory, dependencies, shared RC shell functions, `rc.conf` functionality, etc).
  - designed with a RPM-style script sectioning syntax (e.g. `%start`) and an all-in-one specification approach for seamless integration into the RPM scope.
  - integrates both startup/shutdown (boot!) and periodic (cron!) run-command functionality.
- A run-command script in OpenPKG RPM and RC is always a GNU Bash script, independent of the underlying platform.
- As a result, for a particular packaged application...
  - the OpenPKG RPM package specification covers the build-time and install-time run-commands.
  - the OpenPKG RC package specification covers the run-time run-commands.
- The OpenPKG RC facility consists of:
  - *prefix*`/etc/rc`
  - *prefix*`/etc/rc.func`
  - *prefix*`/etc/rc.conf`
  - *prefix*`/etc/rc.d/rc.`*package*

To me **Vi** is **Zen**. To use Vi is to practice Zen. Every command is a **Koan**. Profound to the user, unintelligible to the uninitiated. You discover truth everytime you use it. — *Achim Bohnet*

Run-Command Facility (2) Gory Details

- Command Line Interface:
  `# openpkg rc` *package command*
- The *package* argument is
  - either `foo` (particular package).
  - or `all` (all packages at once).
- The *command* argument is an arbitrary command orresponding to a "*%command*" section in `rc.`*package*.
- The following commands are well-known and implemented by all packages with `rc.`*package*:
  `status start stop`
- Other well-known sections:
  `restart reload`
  `quarterly hourly daily`
  `weekly monthly`
- Two special sections exist:
  - `%config`: contains defaults for configuration variables which can be overridden from rc.conf
  - `%common`: contains run-commands common to all other sections (except %config)
- Running *prefix*`/bin/openpkg rc foo start` runs a GNU Bash script assembled from
  - `%config` sections from all *prefix*`/etc/rc.d/rc.*`
  - sourcing of *prefix*`/etc/rc.conf`
  - `%common` section from *prefix*`/etc/rc.d/rc.foo`
  - `%start` section from *prefix*`/etc/rc.d/rc.foo`

OSSP fsl (Faking Syslog Library)

- An inherent design goal of OpenPKG is to support multiple instances.
- Major problems with multiple installations of the same application are
  - the listening to the network address/port.
  - the logging via the central syslog(3) facility.
- Conflicts on network listening most of the time can be solved easily by just re-configuring the application.
- Syslog(3) usage in multiple installations of the same application usually results in merged logfile entries in the central logfiles.
- OpenPKG solves the syslog(3) problem with OSSP fsl, a faking syslog(3) library.
- OSSP fsl emulates the syslog(3) API but instead of sending the log message to syslogd(8) it is sending it through a tree of chained channels.
- The tree of chained channels can be configured individually for each application through pattern matching in *prefix*`/etc/fsl/fsl.`*package*.
- OpenPKG by default links all applications using syslog(3) against OSSP fsl and directs their log messages to logfiles staying inside their OpenPKG instance (usually *prefix*`/var/`*package*`/`*package*`.log`)

Part VII: Finish

More about OpenPKG...

The **Apache Group**: a collection of talented individuals who are trying to perfect the art of **never finishing** something.

— Rob Hartill

More about OpenPKG...

- The Website:
  http://www.openpkg.org/
- The FTP Server:
  ftp://ftp.openpkg.org/
- The RSYNC Server:
  rsync://rsync.openpkg.org/
- The CVS Server:
  http://cvs.openpkg.org/
- The OpenPGP Key Server:
  http://pgp.openpkg.org/
  hkp://pgp.openpkg.org/
- The Community Mailing Lists:
  openpkg-announce@openpkg.org
  openpkg-users@openpkg.org
  openpkg-dev@openpkg.org
  openpkg-cvs@openpkg.org

I have made this **longer** than usual because I **lack the time** to make it **shorter**. — *Blaise Pascal*