Unix SW Packaging in the DJB /package Format

3rd party tool independent packaging of Software

Dr. Erwin Hoffmann

<feh@fehcom.de>

FEHCom

20.06.2017

Part I: SW Odyssee

Problem Statement: Upgrade Disaster

Upgrading my FreeBSD 10 to release 11 (using the '`pkg upgrade`' mechanism) was not to successful ....

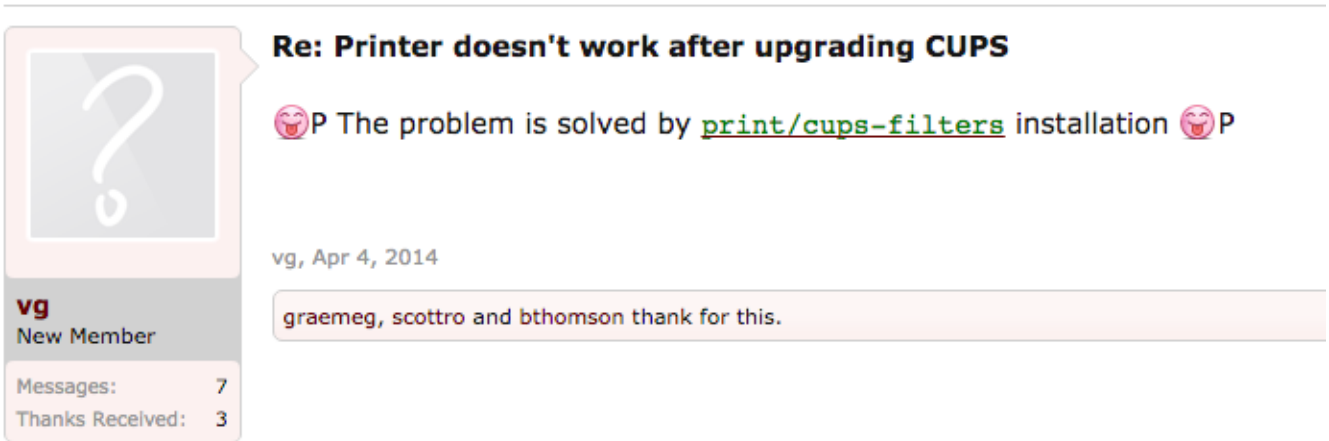
```
[erwin@bigchief ~]$ ls -la pkg*
-rw-------  1 root  users  64241664 Jun  9 19:05 pkg-static.core
```

```
root@bigchief:/usr/lib # ls -la *cryp*
-r--r--r--  1 root  wheel    88908 Jun  9 18:55 libcrypt.a
lrwxr-xr-x  1 root  wheel       23 Jun  9 18:55 libcrypt.so -> ../../lib/libcrypt.so.5
-r--r--r--  1 root  wheel    94086 Jun  9 18:55 libcrypt_p.a
-r--r--r--  1 root  wheel  4364264 Jun  9 18:55 libcrypto.a
lrwxr-xr-x  1 root  wheel       24 Jun  9 18:55 libcrypto.so -> ../../lib/libcrypto.so.8
lrwxr-xr-x  1 root  wheel       24 Jun  9 20:48 libcrypto.so.7 -> ../../lib/libcrypto.so.8
-r--r--r--  1 root  wheel  4615526 Jun  9 18:55 libcrypto_p.a
```

Figure: Shared libs after upgrading from FreeBSD 10 to 11

Problem Statement: Missing Dependencies

A few days later (after a complete re-installation from scratch) CUPS is still not working. I followed this thread: https://forums.freebsd.org/threads/45738/:



Re: Printer doesn't work after upgrading CUPS

:P The problem is solved by print/cups-filters installation :P

vg, Apr 4, 2014

graemeg, scottro and bthomson thank for this.

vg
New Member

Messages: 7
Thanks Received: 3

Figure: Advise to install cups-filter

This makes sense!

Problem Statement: Port Repository dependencies?

If you buy from Amazon USA, please support us by using this linl

Port details

cups-filters Additional backends, filters and other software for CUPS

1.13.5 print Σ=4

Maintainer: cyberbotx@cyberbotx.com

Port Added: 30 Mar 2014 21:34:00

License: BSD4CLAUSE GPLv2 GPLv2+ GPLv3 GPLv3+ LGPL20 LGPL21+ MIT

```
The CUPS Filters package contains backends, filters and other
software that was once part of the core CUPS distribution but is
no longer maintained by Apple Inc.

WWW: http://www.linuxfoundation.org/collaborate/workgroups/openprinting/cups-filters
```

SVNWeb : Homepage : PortsMon

To install the port: `cd /usr/ports/print/cups-filters/ && make install clean`
To add the package: `pkg install cups-filters`

PKGNAME: cups-filters

distinfo:

```
TIMESTAMP = 1493593539
SHA256 (cups-filters-1.13.5.tar.xz) = 35db1c5821c9ff0e0fedcf87b3ae68a424ad951bd8af421a2a1aac5613e17b8d
SIZE (cups-filters-1.13.5.tar.xz) = 1424764
```

Figure: FreeBSD 11 cups-filter Repository

Problem Statement: Dependency Hell

```
cd /usr/ports/printer/cups-filter
make
```

```
Jun 14 15:25:38 bigchief pkg-static: dialog4ports-0.1.6 installed
Jun 14 15:28:53 bigchief pkg-static: gettext-tools-0.19.8.1 installed
Jun 14 15:29:01 bigchief pkg-static: gmake-4.2.1_1 installed
Jun 14 15:29:08 bigchief pkg-static: pkgconf-1.3.7,1 installed
Jun 14 15:44:05 bigchief pkg-static: lcms2-2.8 installed
Jun 14 15:44:54 bigchief pkg-static: zip-3.0_1 installed
Jun 14 15:45:24 bigchief pkg-static: nspr-4.15 installed
Jun 14 15:46:55 bigchief pkg-static: sqlite3-3.19.3 installed
Jun 14 15:49:05 bigchief pkg-static: nss-3.31 installed
Jun 14 15:50:55 bigchief pkg-static: py27-MarkupSafe-1.0 installed
Jun 14 15:51:15 bigchief pkg-static: py27-pytz-2016.10,1 installed
Jun 14 15:51:17 bigchief pkg-static: py27-Babel-2.3.4 installed
Jun 14 15:51:19 bigchief pkg-static: py27-Jinja2-2.9.5 installed
Jun 14 15:51:28 bigchief pkg-static: py27-docutils-0.13.1 installed
Jun 14 15:51:30 bigchief pkg-static: py27-six-1.10.0 installed
Jun 14 15:51:40 bigchief pkg-static: py27-pygments-2.2.0 installed
Jun 14 15:51:45 bigchief pkg-static: py27-sphinx_rtd_theme-0.2.4 installed
Jun 14 15:51:47 bigchief pkg-static: py27-alabaster-0.7.6 installed
Jun 14 15:51:56 bigchief pkg-static: py27-pystemmer-1.3.0_1 installed
Jun 14 15:51:57 bigchief pkg-static: py27-snowballstemmer-1.2.0_1 installed
Jun 14 15:52:00 bigchief pkg-static: py27-imagesize-0.7.1 installed
Jun 14 15:52:03 bigchief pkg-static: py27-sphinx-1.4.8_1,1 installed

.......
```

Problem Statement: Dependency Hell /2

```
.......

Jun 14 15:52:44 bigchief pkg-static: p5-Locale-gettext-1.07 installed
Jun 14 15:52:46 bigchief pkg-static: help2man-1.47.4 installed
Jun 14 15:53:28 bigchief pkg-static: texinfo-6.3_2,1 installed
Jun 14 15:53:57 bigchief pkg-static: m4-1.4.18,1 installed
Jun 14 15:54:12 bigchief pkg-static: autoconf-wrapper-20131203 installed
Jun 14 15:54:12 bigchief pkg-static: autoconf-2.69_1 installed
Jun 14 15:54:23 bigchief pkg-static: automake-wrapper-20131203 installed
Jun 14 15:54:24 bigchief pkg-static: automake-1.15_1 installed
Jun 14 15:54:34 bigchief pkg-static: libtool-2.4.6 installed
Jun 14 15:55:00 bigchief pkg-static: libnghttp2-1.23.1 installed
Jun 14 15:56:01 bigchief pkg-static: curl-7.54.0 installed
Jun 14 15:56:13 bigchief pkg-static: scons-2.5.1_1 installed
Jun 14 15:56:27 bigchief pkg-static: jsoncpp-1.8.0_2 installed
Jun 14 15:56:51 bigchief pkg-static: libuv-1.12.0 installed
Jun 14 15:57:21 bigchief pkg-static: rhash-1.3.4 installed
Jun 14 16:05:58 bigchief pkg-static: cmake-modules-3.8.2 installed
Jun 14 16:06:12 bigchief pkg-static: cmake-3.8.2 installed
Jun 14 16:06:26 bigchief pkg-static: openjpeg-2.1.2_1 installed

.......
```

Problem Statement: Dependency Hell /3

```
.......

Jun 14 16:07:26 bigchief pkg-static: poppler-data-0.4.7 installed
Jun 14 16:07:27 bigchief pkg-static: poppler-0.50.0 installed
Jun 14 16:07:49 bigchief pkg-static: poppler-glib-0.50.0 installed
Jun 14 16:08:03 bigchief pkg-static: poppler-utils-0.50.0_1 installed
Jun 14 16:38:37 bigchief pkg-static: jbig2dec-0.13 installed
Jun 14 16:38:54 bigchief pkg-static: svgalib-1.4.3_7 installed
Jun 14 16:40:48 bigchief pkg-static: gsfonts-8.11_8 installed
Jun 14 16:41:01 bigchief pkg-static: ghostscript9-agpl-base-9.16_5 installed
Jun 14 16:42:21 bigchief pkg-static: qpdf-6.0.0_1 installed
Jun 14 16:42:28 bigchief pkg-static: libijs-0.35_5 installed
Jun 14 16:44:43 bigchief pkg-static: jam-2.6 installed
Jun 14 16:47:17 bigchief pkg-static: argyllcms-1.9.2_1 installed
Jun 14 16:47:23 bigchief pkg-static: p5-XML-Parser-2.44 installed
Jun 14 16:47:25 bigchief pkg-static: intltool-0.51.0_1 installed
Jun 14 16:50:58 bigchief pkg-static: spidermonkey170-17.0.0_6 installed
Jun 14 16:51:15 bigchief pkg-static: polkit-0.113_5 installed
Jun 14 16:52:18 bigchief pkg-static: colord-1.2.12 installed

Jun 14 16:52:19 bigchief dbus[742]: [system] Reloaded configuration
Jun 14 17:02:51 bigchief pkg-static: cups-filters-1.13.5 installed
```

- End: 17:02:51 (including down-load times)

Problem Statement: Dependency Hell /3

```
.......

Jun 14 16:07:26 bigchief pkg-static: poppler-data-0.4.7 installed
Jun 14 16:07:27 bigchief pkg-static: poppler-0.50.0 installed
Jun 14 16:07:49 bigchief pkg-static: poppler-glib-0.50.0 installed
Jun 14 16:08:03 bigchief pkg-static: poppler-utils-0.50.0_1 installed
Jun 14 16:38:37 bigchief pkg-static: jbig2dec-0.13 installed
Jun 14 16:38:54 bigchief pkg-static: svgalib-1.4.3_7 installed
Jun 14 16:40:48 bigchief pkg-static: gsfonts-8.11_8 installed
Jun 14 16:41:01 bigchief pkg-static: ghostscript9-agpl-base-9.16_5 installed
Jun 14 16:42:21 bigchief pkg-static: qpdf-6.0.0_1 installed
Jun 14 16:42:28 bigchief pkg-static: libijs-0.35_5 installed
Jun 14 16:44:43 bigchief pkg-static: jam-2.6 installed
Jun 14 16:47:17 bigchief pkg-static: argyllcms-1.9.2_1 installed
Jun 14 16:47:23 bigchief pkg-static: p5-XML-Parser-2.44 installed
Jun 14 16:47:25 bigchief pkg-static: intltool-0.51.0_1 installed
Jun 14 16:50:58 bigchief pkg-static: spidermonkey170-17.0.0_6 installed
Jun 14 16:51:15 bigchief pkg-static: polkit-0.113_5 installed
Jun 14 16:52:18 bigchief pkg-static: colord-1.2.12 installed

Jun 14 16:52:19 bigchief dbus[742]: [system] Reloaded configuration
Jun 14 17:02:51 bigchief pkg-static: cups-filters-1.13.5 installed
```

- End: 17:02:51 (including down-load times)
- Start: 15:25:38

Problem Statement: Tool Hell

A standard Unix SW installation from the sources just needs a 'C' compiler (**cc**) and a linker **ld**. Here, we have two choices:

- The traditional GNU environment: gcc[1] and binutils[2].
- The BSD-oriented path: clang[3] and LLVM[4].

Apart from that, the Makefile[5] utility is indispensable.

↪ This is enough, to build (and distribute) source code for a large variety of the POSIX conforming Operating Systems.

Apart from the original **make** written by *Stuart Feldman* back in 1976, in particular **gmake** is widely spread. Sun invented **dmake** to build *Open Office* initially.

[1] https://gcc.gnu.org
[2] https://www.gnu.org/software/binutils/
[3] http://clang.llvm.org
[4] http://llvm.org
[5] https://www.gnu.org/software/make/manual/make.html

Problem Statement: Tool Hell .. However ...

Larger SW projects tend to use their own tailored environment.

- Originated by *David Mackenzie* of the FSF, the **autoconf**[6] tool was developed and used by most GNU SW.
- The other major player is **cmake**, used by *MySQL* and *CuRL*. In addition, **clang** and **LLVM** support it[7].
- **imake** has been used in particular for building the X-Window system.

However, today's approach[8] is to provide

- Continuous integration tools,
- Configuration management tools,

for compiled binary programs instead of the source code.

[6] `http://www.gnu.org/software/autoconf/autoconf.html`
[7] `http://llvm.org/docs/CMake.html`
[8] `https://en.wikipedia.org/wiki/Build_automation`

Problem Statement: ./configure Hell

The **autoconf** tool makes use of `M4` macro:

- The *developer* specifies the requirements of his SW in M4 macros, provided in the file `configure.ac`.
- The user call prior of compiling the command **./configure** which actually tries to re-construct the developer's environment on the local Unix machine.
- Now, the final `Makefile` is built and the user is able to compile the SW on his system[9].

↪ At the end of the `./configure` process, the developer's environment is re-established at the user's system. This step is not only error-prone, but in addition takes occasionally much longer than the compile step itself.

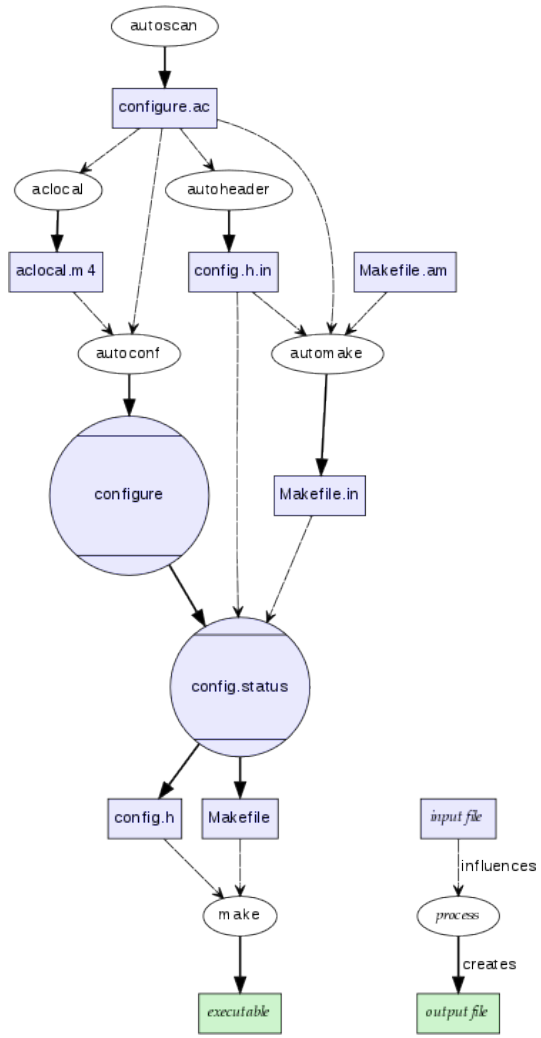


Figure: The Autoconf path [https://en.wikipedia.org/wiki/Autoconf]

[9] http://freecode.com/articles/stop-the-autoconf-insanity-why-we-need-a-new-build-system

Problem Statement: Updates of Servers

A.) Consider the update of a busy **Apache** or **nginx** on a busy server. How to proceed in case you need to install the software from the sources?

- Download the tar-ball.
- Extract the tar-ball under /user/local/src.
- Run **./configure** (cross your fingers and wait).
- Run **make**.
- Shutdown your Web server (takes a while; still busy).
- Run **install** (to deploy the binaries).
- Ups! Daemon is not coming up; same changes in the `conf` script.
- Adjustments doing a vi-session; restart.

B.) Consider updated via a package-manager:

- `apt-get` install ....
- ....
- ....
- Shit, `conf` files have significantly changed; need to go the docu first.

Step 2: Calypso

Solution Statement: The Developer is responsible!

Using the Autoconf, the Developer puts the burden to this tools, hoping it solves the dependencies for the user.

Responsible SW (™) however,

- should have only minimal dependencies w.r.t. the installed system,
- thus provides the basic SW libs 'in-line',
- does not compromise the user's security context,
- allows and easy update/upgrade and downgrade path,
- permits the user to remove all remnants from the system,
- makes installation decisions explicit and transparent,

One solution to cope this these requirements is Dan Bernstein's (DJB) */package* (slashpackage) format.

> Unfortunately, apart from the web site `http://cr.yp.to/slashpackage.html`, there is no description about the (internal) package format.

(1.) /package: Name Allocation

Names are an essential part of the */package* format:

- Each *package* must be part of (sub)category,
- a particular *name* needs to be allocated in a *hierarchy* and
- has to posses relative uniqueness.
- Maintainer of the */package* tree is DJB; mainly to avoid name clashes.

Allocated categories:

- `admin/` for system administration.
- `data/` for databases, data structures, etc.
- `mail/` for Internet mail, instant messaging, etc.
- `map/` for street maps, weather maps, star maps, etc.
- `math/` for mathematics.
- `misc/` for miscellany.
- `prog/` for general programming.
- `net/` for network communication.
- `text/` for text editors, text processors, etc.
- `web/` for network publication: browsers, HTTP servers, FTP servers, etc.

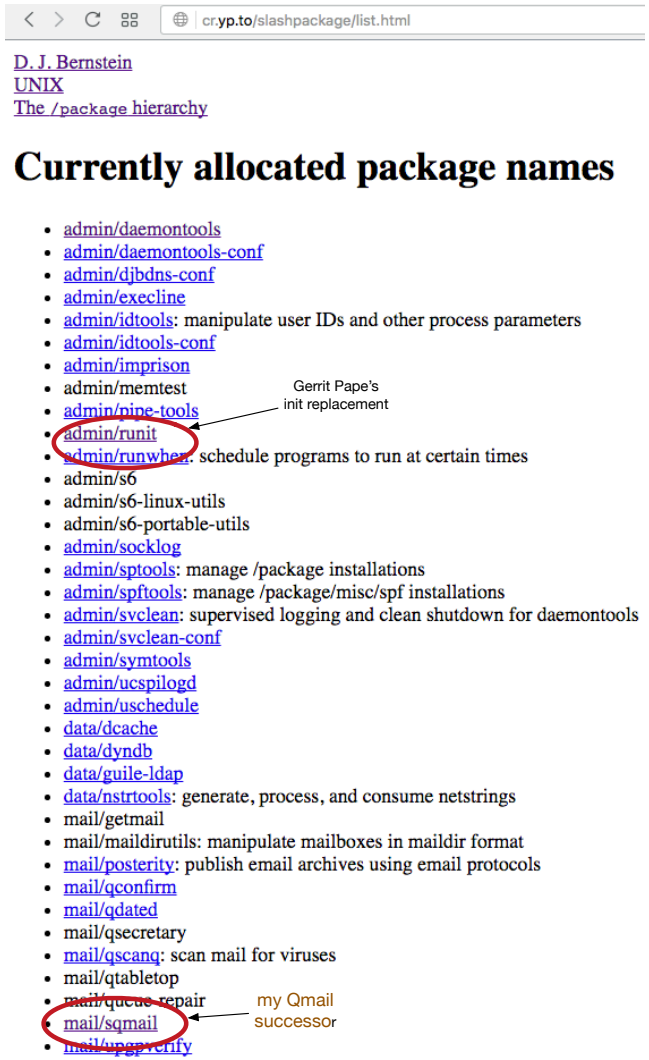


Figure: Currently allocated */package* space (partially)

(2.) /package: Directory Structure

The given package name is automatically provisioned in the directory hierarchy:

1. `mkdir /package; chmod 1777 /package`
2. `cd /package; tar -xzf PATH/package.tgz`
3. `cd /package/CATEGORY/SUBCATEGORY/PACKAGENAME`

```
1 ls -la /package
  total 24
3 drwxr-xr-x  6 root root 4096 May 20  2015 .
  drwxr-xr-x 27 root root 4096 Feb 23 08:42 ..
5 drwxr-xr-t  3 root root 4096 Jan 17  2014 admin
  drwxr-xr-x  7 root root 4096 Mar  1 22:59 host
7 drwxr-xr-x  3 root root 4096 May 20  2015 mail
  drwxr-xr-x  2 root root 4096 Apr 15  2014 net
```

Listing 1: /package directory hierarchy

↪ Thus, the tar-ball needs to carry the respective information; the user not required to raise directories in advance.

(3.) /package: Automatic Versioning

Each package should be constructed such,

a) it is provided as a tar-ball,

b) carries an version identifier: `packagename-VRF.tgz`,

c) upon un-taring the tar-ball, the created directory shall have the same name.

```
cd /package/mail/sqmail; ls
sqmail          sqmail-2.13.16 sqmail-2.14.04 sqmail-3.0.0 sqmail-3.1.6  sqmail-3.2.17
sqmail-2.11.3   sqmail-2.13.19 sqmail-2.14.09 sqmail-3.10  sqmail-3.1.7  sqmail-3.2.18
sqmail-2.12.13  sqmail-2.13.21 sqmail-2.14.13 sqmail-3.1.0 sqmail-3.1.8  sqmail-3.2.19
sqmail-2.12.14  sqmail-2.13.4  sqmail-2.15.11 sqmail-3.1.3 sqmail-3.1.9  sqmail-3.3.3
sqmail-2.12.9   sqmail-2.14.02 sqmail-2.16.06 sqmail-3.1.4 sqmail-3.2.14
sqmail-2.13.0   sqmail-2.14.03 sqmail-2.17.02 sqmail-3.1.5 sqmail-3.2.15
```

Listing 2: Automatic package versioning

The CURRENT version is provided as *symlink* automatically upon installation:

```
lrwxrwxrwx  1 root root   12 Jun 15 13:52 sqmail -> sqmail-3.3.3
drwxr-xr-x 13 root root 4096 May 24  2015 sqmail-2.11.3
...
drwxr-xr-x 13 root root 4096 Apr  8 17:07 sqmail-3.2.19
drwxr-xr-x 13 root root 4096 Jun 15 13:58 sqmail-3.3.3
```

Listing 3: Current version of package

(4.) /package: Installation

The idea of */package* is simple:

- Upon tar-ball extraction, **cd** to the generated directory.
- Run `./package/`**install**

A minimal *package* consists of just two directories:

1. The `./src` directory containing the ... source files.
2. The `./package` directory includes (at least) the **compile** and **install** script.

↪ The user does not need to do any '**./configure**'; the developer is responsible

- to detect the OS and CPU,
- include perhaps creating of required *userid* and *groups*,
- to identify the required shared resources of the OS and include them.

The compile script will raise a `./compile` directory as link to the `./src` directory and the entire compilation (running **make**) is done in this directory (hosting all the generated artifacts and binaries).

(5.) /package: Executable Provisioning

The interface with the Unix Operating System is indirect:

- Upon package installation a `./command` directory is generated used as container for all executables.
- From here, symlinks are provided to point (for each file individually) to typical `/usr/local/bin`.

```
cd /usr/local/bin
ls -la tcp*
lrwxrwxrwx 1 root staff 39 Mar  1 22:59 tcpcat -> /package/host/ucspi-tcp6/command/tcpcat
lrwxrwxrwx 1 root staff 42 Mar  1 22:59 tcpclient -> /package/host/ucspi-tcp6/command/tcpclient
lrwxrwxrwx 1 root staff 41 Mar  1 22:59 tcprules -> /package/host/ucspi-tcp6/command/tcprules
lrwxrwxrwx 1 root staff 46 Mar  1 22:59 tcprulescheck -> /package/host/ucspi-tcp6/command/tcprulescheck
lrwxrwxrwx 1 root staff 42 Mar  1 22:59 tcpserver -> /package/host/ucspi-tcp6/command/tcpserver
```

Listing 4: Nesting of executables

↪ Here, always the CURRENT version of the package is used as source for symlink: Double symlinking!

Step 3: Polyphem

The blind(ed) Giant

Dan Bernstein has just provided some samples how to generated package tar-balls, thus the 'magic' to generate package files needs to be reversed-engineered and tailored to the specific demands.

Idea:

- (Infrastructure) Each package is generated and maintained exactly as is should be un-tared at it's destination.
- (Infrastructure) The package's name and the version number has to be included in the directory name.
- (Build process) The PREFIX (i.e. `/package/mail/`) is added as part of the 'build' process.
- (Build process) The build process is responsible to guarantee completeness and soundness of the generated package.

Polyphem: Infrastructure

Each package I generate, provides the following (sub) directories:

- `./package` – installation scripts and maintenance files; hook to build.
- `./src` – the source files; including Makefile and abstraction of targets.
- `./man` – the man files (including Makefile).
- `./doc` – additional documentation.
- `./etc`, `./service`, `./scripts` – additional run-time configuration files.

Apart form those subdirectories, on the top-level you will find the following (mostly configuration) files:

- README – blurb about the package.
- `conf-cc` – adjustments for compiler settings; if not automatically detected.
- `conf-ld` – adjustments for loader settings; if not automatically detected.
- `conf-XX` – additional configuration files.

↪ Executing **package/compile** (and other scripts), these `conf`-files are **sed**'ed to the sources (i.e. Makefile).

Polyphem: The package dir

The package dir for each package follows the definitions of Daniel Bernstein (which occasionally is a mystery to me).

In the simple case of the `ucspi-tcp6` package, the package directory includes the following files:

```
-rw-rw-r--  1 ucspi  users    92 Aug 19  2012 command-cp
-rw-rw-r--  1 ucspi  users   112 Apr 15  2014 command-ln
-rw-rw-r--  1 ucspi  users   140 May 26  2013 commands-base
-rwxrwxr-x  1 ucspi  users  1538 Aug 19  2012 compile
-rw-rw-r--  1 ucspi  users  4346 Aug  2  2016 files
-rwxrwxr-x  1 ucspi  users   109 Aug 19  2012 install
-rwxrwxr-x  1 ucspi  users  1248 Aug  2  2016 man
-rw-rw-r--  1 ucspi  users     5 Apr 14  2014 path
-rwxrwxr-x  1 ucspi  users   250 Aug 21  2012 report
-rwxrwxr-x  1 ucspi  users  1322 Aug 19  2012 rts
-rwxrwxr-x  1 ucspi  users    20 Aug 19  2012 run
-rwxrwxr-x  1 ucspi  users  1947 Aug 19  2012 upgrade
```

Listing 5: Nesting of executables

We recognize (by means of the executable bit) the files:

- **compile** – the compile script (generic)
- **man** – the script to generate the man pages (generic)
- **run** – generic run script (generic)
- **install** – the install script (mostly generic)
- **upgrade** – upgrade script from previous installed version (generic)
- **report** – script to email OS parms to me (generic)
- **rts** – real-time test script (not always working)

Polyphem: Make it generic!

One cornerstone of the /package mechanism is, to make everything as *generic* as possible:

```
#!/bin/sh -e
package/compile ${1+"$@"}
package/upgrade ${1+"$@"}
package/man ${1+"$@"}
package/run ${1+"$@"}
```

Listing 6: the package/install script

However, sometimes to be generic needs a good abstraction and ends up in a significant complexity:

```
#!/bin/sh
shout() { echo "compile: $@" >&2; }
barf() { shout "fatal: $@"; exit 111; }
safe() { "$@" || barf "cannot $@"; }
umask 022
[ -d package ] || barf "no package directory"
[ -d src     ] || barf "no src directory"

here='env - PATH=$PATH pwd'

safe mkdir -p compile command
[ -r compile/home ] || echo $here > compile/home
[ -h compile/src  ] || safe ln -s $here/src compile/src

for i in 'ls src'
do
  [ -h compile/$i ] || safe ln -s src/$i compile/$i
done

(60 more lines)
```

Listing 7: The package/compile script

Polyphem: The package installation dirs

Typically, the executable files are provided as symlinks. Responsible is the configuration file `command-ln`:

```
/usr/local/bin

Directories to soft link commands into, one per line.
The first empty line terminates the list.
```

Listing 8: The package/command-ln file

Using the file `command-cp`, tells /package to copy the executables (additionally) to a specific target directory:

```


Directories to copy commands into, one per line.
The first empty line terminates the list.
```

Listing 9: The package/command-cp file

↪ The first line of the script is again used as the input for a **head** command.

Polyphem: Tagging a package build

Generating a new package (via the 'build' script')

- will automatic generate the `/package` PREFIX for the package (by means of the provided `./package/path`),
- will include the `./package/version` number (from the directory name)
- will include a `./package/build` number (as timestamp)

The tar-ball is generated with these 'build' information and a MD5 hash shum is calculated for reference.

Polyphem: The package installation completeness

Within the directories of the specific package, we will find all kinds of files; partially artifacts from tests. To include those in the tar-ball is ugly, but usually does not harm. However, a missing file would probably be a *installation-break*.

The file `./package/files` lists all files to be required for provisioning:

```
conf-cc
conf-ld
conf-man
README
doc
doc/CHANGES
doc/INSTALL
doc/LICENSE
doc/TODO
man
man/addcr.1
man/argv0.1
man/date@.1
man/delcr.1
```

Listing 10: The package/command-ln file

↪ The 'build' script verifies that all files are included. Missing and obsolete files are indicated.

Polyphem: The package installation soundness

The Makefile utility provides a scheme known as 'targets':
→ You can specify selective compiling for a set of objects.

The /package mechanism supports this, however requiring a hook between the 'package' and the 'src' information:

- Commands can be grouped and indicated on the package dir in files named `commands-XX`.
- These commands are 'targeted' in the src directory as `./src/it-XX`.
- Several command groups are stacked together in the file `./src/id=d`.
- Within the Makefile, these targets need to be defined.

```
# Don't edit Makefile! Use ../conf-* for configuration.
SHELL=/bin/sh
default: it
it: it-base sysdeps
it-base: \
tcpserver tcprules tcprulescheck argv0 recordio tcpclient who@ date@ \
finger@ http@ tcpcat mconnect mconnect-io addcr delcr fixcrio rblsmtpd \
sysdeps

load: \
warn-auto.sh ../conf-ld
        ( cat warn-auto.sh; \
        echo 'main="$$1"; shift'; \
        echo exec "`head -1 ../conf-ld`" \
        '-o "$$main" "$$main".o $${1+"$$@"}' \
        ) > load
        chmod 755 load
```

Listing 11: Makefile with targets

Polyphem: Redoing your work

In case, the compilation does not succeed don't worry:

- `./src` directory and `./compile` directory are decoupled.
- Remove the `./compile` directory and do it again.
- In my packages, 'make clean' is provided as well.

Polyphem: Identifying OS and resources

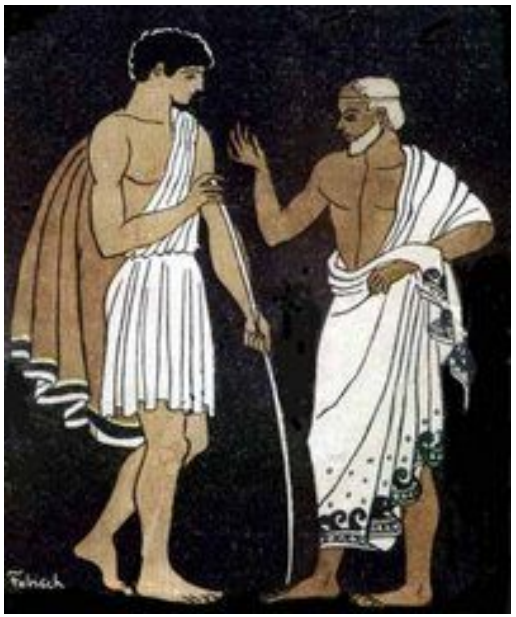
- The developer is required to provide test programs (within ./src) to check prior of compiling the ability of the OS.
- This is usually done by means of some test scripts called in the 'sysdeps' phase during Makefile execution and generating the required information for the following compile steps.

```
ld="`head -1 ../conf-ld`"
systype="`cat systype`"

flag=0

rm -f trycpp.o

flag=`cc -c tryssl.c -m64 2>&1 | wc -l`
if [ $flag -eq 0 ]; then
  ld="$ld -m64"
fi

rm -f trycpp.o

cat warn-auto.sh
echo 'main="$1"; shift'
echo exec "$ld" '-o "$main" "$main".o ${1+"$@"}'
```

Listing 12: Detecting 64 bit OS

↪ These kind of dependencies handling works even for *multi-threading* compilations.

Step 4: Telemach



Telemach: Let's go for installation!

Questions?

Comments?

Do it!

Resources and further information


Daniel J. Bernstein
The /package hierarchy
http://cr.yp.to/slashpackage.html


Tom Preston-Werner
Semantic Versioning
http://semver.org


David MacKenzie and Ben Elliston
Autoconf - Creating Automatic Configuration Scripts
ftp://ftp.gnu.org/old-gnu/Manuals/autoconf-2.13/html_chapter/autoconf_toc.html