ASDF 3

Why Lisp is Now an Acceptable Scripting Language

François-René Rideau <tunes@google.com>

European Lisp Symposium, 2014-05-05

ASDF 3

Another System Definition Facility

build system

de facto standard

DEFSYSTEM tradition

in-image

version 3

Brief History of ASDF

Prehistory: load scripts

1970s DEFSYSTEM

1990 MK-DEFSYSTEM

2002 ASDF (Dan Barlow ≠ me)

2010 ASDF 2 (me me!)

2013 ASDF 3 (me me me!)

An Acceptable Scripting Language

Writing a Unix-style script in Lisp

```
#!/usr/bin/cl -sp lisp-stripper -E main
(defun main (argv)
  (if argv
      (map () 'print-loc-count argv)
      (print-loc-count *standard-input*)))
```

*lispwc *.lisp*

Invoking Lisp code from the shell

```
#!/bin/sh

form='`#5(1 ,@`(2 3))'

for impl in allegro ccl clisp sbcl ecl \
            lispworks abcl cmucl gcl scl xcl ; \
do
  cl -l $impl \
      "(format t \"$impl ~S~%\" $form)" \
  2>&1 | grep "^$impl " # LW, GCL are verbose
done
```

Invoking external commands from Lisp

```
#!/usr/bin/cl -sp inferior-shell

(loop with form = "`#5(1 ,@`(2 3))"

   for impl in '(allegro ccl clisp sbcl ecl
                 lispworks abcl cmucl gcl scl xcl)
   do
   (run `(pipe (cl -l ,impl (>& 2 1)
                    ("(format t \"" ,impl " ~S~%\" "
                       ,form ")"))
            (grep ("^" ,impl " "))))))
```

Better abstractions for scripting

```
(loop with form = "`#5(1 ,@`(2 3))"

   for impl in '(allegro ccl clisp sbcl ecl
                 lispworks abcl cmucl gcl scl xcl)
   collect
   (run `(pipe (cl -l ,impl (>& 2 1)
                  ("(format t \"" ,impl " ~S~%\" "
                     ,form ")"))
            (grep ("^" ,impl " "))) :output :forms))
```

Standards-based portability

```
`#5(1 ,@`(2 3))
```

```
((ALLEGRO #(1 2 3 2 3 2 3 2 3))
 (CCL #(1 2 3 2 3 2 3 2 3)))
 (CLISP #(1 2 3 2 3 2 3 2 3)))
 (SBCL #(1 2 3 2 3 2 3 2 3)))
 (ECL #(1 2 3 3 3)))
 (LISPWORKS #(1 2 3 3 3)))
 (ABCL #(1 2 3)))
 (CMUCL #(1 2 3)))
 (GCL #(1 2 3)))
 (SCL #(1 2 3)))
 (XCL #(1 2 3))))
```

What prevented scripting?

What prevented scripting?

(No) Write Once, Run Most-anywhere

What prevented scripting?

finding source code

locating output files

command line invocation

argv access

run-program

pipes, expansion

What made scripting possible?

finding source code → asdf2 (source-registry)

locating output files → asdf2 (output-translations)

command line invocation → cl-launch

argv access → cl-launch

run-program → asdf3 (uiop)

pipes, expansion → inferior-shell

Finding source code (before)

Q: Where is system **`foo`** ?

The hard way: modify every client

logical-pathname: system and client must agree

ASDF 1: user maintains a link farm to .asd files

But how to configure? **`~/.sbclrc`**, etc.

Finding source code (after)

ASDF 2: **`source-registry`**

Implementation-independent

Nice DSL

Can recurse into subtrees

Prog > Env > User > Sys, Explicit > Defaults

Sensible defaults

ASDF 3.1: **`~/common-lisp/`**

Finding source code (lessons)

Who knows specifies, who doesn't needn't

It *just works* by default

Modular configuration

Reusable DSL for pathname designators

Better than in C!

Locating output files (before)

Output in source code directory

(Technically as bad as for C)

no standard paths, no common ABI

(Socially worse than for C)

Locating output files (before)

Output in source code directory

(Technically as bad as for C)

no standard paths, no common ABI

(Socially worse than for C)

Libraries as source (like Perl…)

Cannot share source code

ASDF 1: output redirection, but...

`defmethod output-files :around (o c)`

c-l-c (2002), A-B-L (2005)

ASDF 1: output redirection, but...

`defmethod output-files :around (o c)`

c-l-c (2002), A-B-L (2005)

Not modular

Where to configure? **`~/.sbclrc`**

Locating output files

ASDF 2: **`output-translations`**

Configuration similar to **`source-registry`**

Before: as bad as C, but without conventions

Default: persistent cache, per user, per ABI

Cache not shared, for security

a JIT, but persistent and coarse-grained

a portable bytecode VM, with code 40, 41

Shell interface

shell-to-Lisp: **`cl-launch`**

Lisp-to-shell: **`uiop/run-program`**, **`inferior-shell`**

100% solution, 100% portable

Related Improvements

Easier delivery with bundle operations

Deliver an executable: **`cl-launch`**

Deliver a library: **`asdf:compile-bundle-op`**

Deliver code as only one or two files!

Image Life-cycle support

Need to use environment variables?

```
(uiop:register-image-dump-hook 'clear-env-vars)

(uiop:register-image-restore-hook 'init-env-vars)
```

Image Life-cycle support

Need to use environment variables?

```
(uiop:register-image-dump-hook 'clear-env-vars)

(uiop:register-image-restore-hook 'init-env-vars)
```

Many other uses

A standard interface *matters*

Scripting Language?

Scripting Language?

Low-overhead programming

No boilerplate

Write once, run everywhere *unmodified*

No setup needed

Spawn or be spawned by other programs

call or be called by functions in other languages

What is it all about?

What is it all about?

ASDF 3 does nothing that cannot be done without it

What is it all about?

ASDF 3 does nothing that cannot be done without it

Neither does any piece of software

What is it all about?

ASDF 3 does nothing that cannot be done without it

Neither does any piece of software

Division of labor

What is it all about?

ASDF 3 does nothing that cannot be done without it

Neither does any piece of software

Division of labor

Enabling the division of labor

Beyond ASDF 3

Beyond ASDF 3

less overhead:

ASDF 3.1: **`asdf:package-inferred-system`**

more modularity:

ASDF 3.1: **`*readtable*`** protection

more access:

Integration with other languages?

Lessons for other languages

less overhead

more modularity

more access

Also in the extended article...

The basic design of ASDF

Why it rocks / sucks compared with C build tools

Innovations in ASDF 1 2 2.26 3 3.1

The Problem with Pathnames

Lessons in Software Design including Pitfalls

A great bug chase story

http://github.com/fare/asdf3-2013

Share and Enjoy!

http://common-lisp.net/project/asdf/

http://cliki.net/cl-launch

http://cliki.net/inferior-shell

http://www.quicklisp.org/beta/

http://github.com/fare/asdf3-2013

Any Questions?