How to save the environment

..and get rid of virtualenv, rvm, pythonbrew, rbenv, pythonz (...)

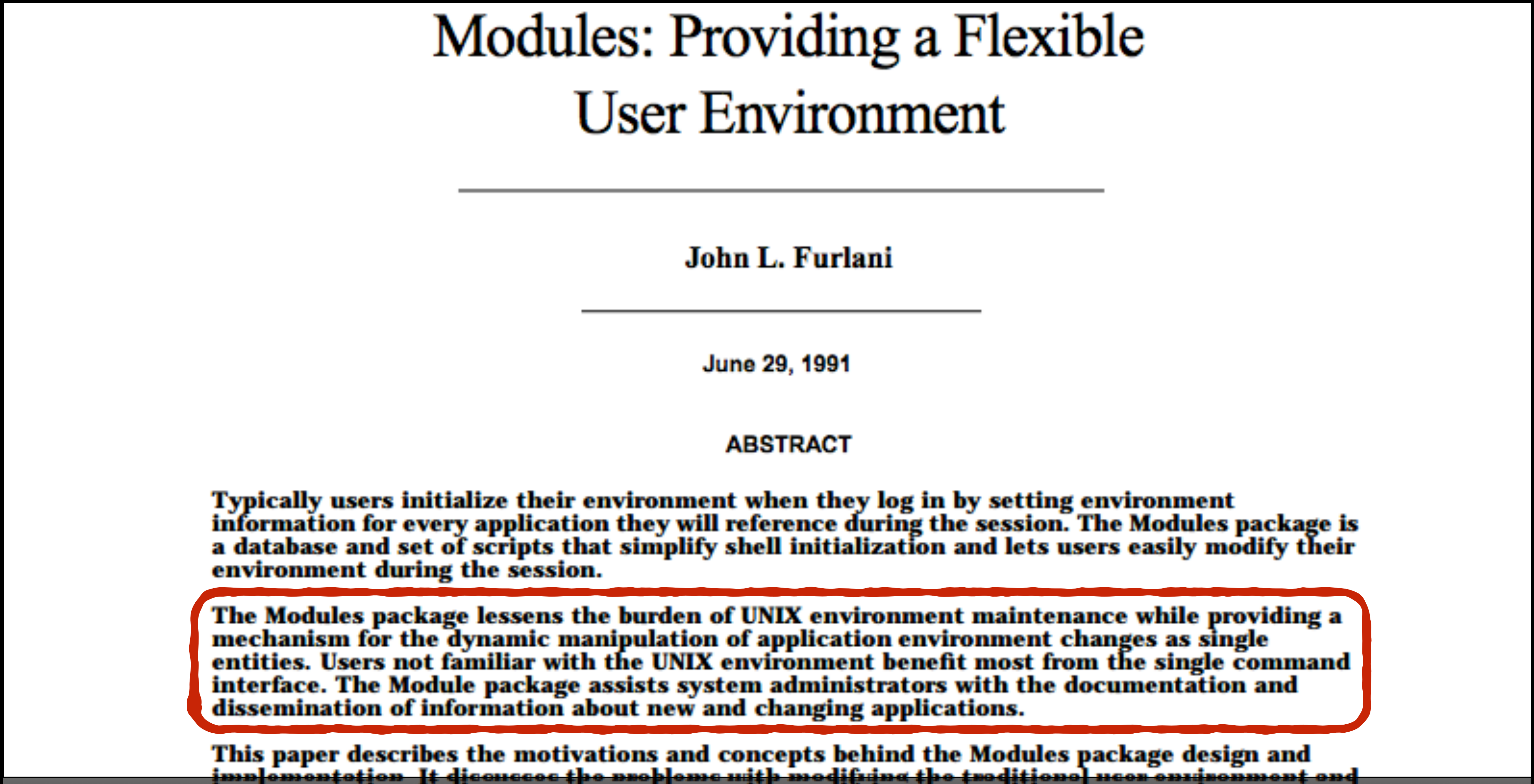
FOSDEM 2014 - Brussels, Belgium, Old World
Earth, Orion–Cygnus Arm, Milky Way

Aaron Zauner
azet@azet.org
@a_z_e_t

`whoami`

- Self-employed engineer currently involved in
  - IT-Infrastructure, HPC and security stuff
  - primarily working for biology institutes of the austrian academy of sciences
- Used to do
  - Web/Mail and Datacenter Operations
  - Front/Backend Dev.
  - Security auditing
  - FOSS/Community (BSD)
  - […]

Slide intentionally left blank for your enjoyment.



Modules: Providing a Flexible User Environment

John L. Furlani

June 29, 1991

ABSTRACT

Typically users initialize their environment when they log in by setting environment information for every application they will reference during the session. The Modules package is a database and set of scripts that simplify shell initialization and lets users easily modify their environment during the session.

The Modules package lessens the burden of UNIX environment maintenance while providing a mechanism for the dynamic manipulation of application environment changes as single entities. Users not familiar with the UNIX environment benefit most from the single command interface. The Module package assists system administrators with the documentation and dissemination of information about new and changing applications.

This paper describes the motivations and concepts behind the Modules package design and implementation. It discusses the problems with modifying the traditional user environment and

‘Environment Modules’

First officially published at LISA V back in 1991 by John L. Furlani

Environment Modules

- In use ever since in High Performance and Scientific Computing
- Ease of administration
- Ease of use

Environment Modules

- by most just refered to as “Modules”
- originally intended to enable multiple users to have software in different versions available
- traditionally most HPC environments do not use system packages
- special compilers, toolchains, math libraries, optimisation, interconnect-specifics […]
- and often scientific software is not to be found in $distro repo (because it’s crap || not used outside the sci. community)
- somewhat like Gentoo or Arch - but a lot more cumbersome for sysadmins and users
- a system to manage multiple versions of the same software with different toolchains is needed that is easy to maintain
- and easy for users (developers and scientists, usually) to learn and use

Environment Modules

- “modulefiles” dynamically set an environment specific to a software
- may load/unload other modulefiles if there is a dependency present
- examples include but are not limited to
  - PATH
  - MANPATH
  - LD_LIBRARY_PATH
  - LD_PRELOAD

Environment Modules

(with Lmod - more on that later)

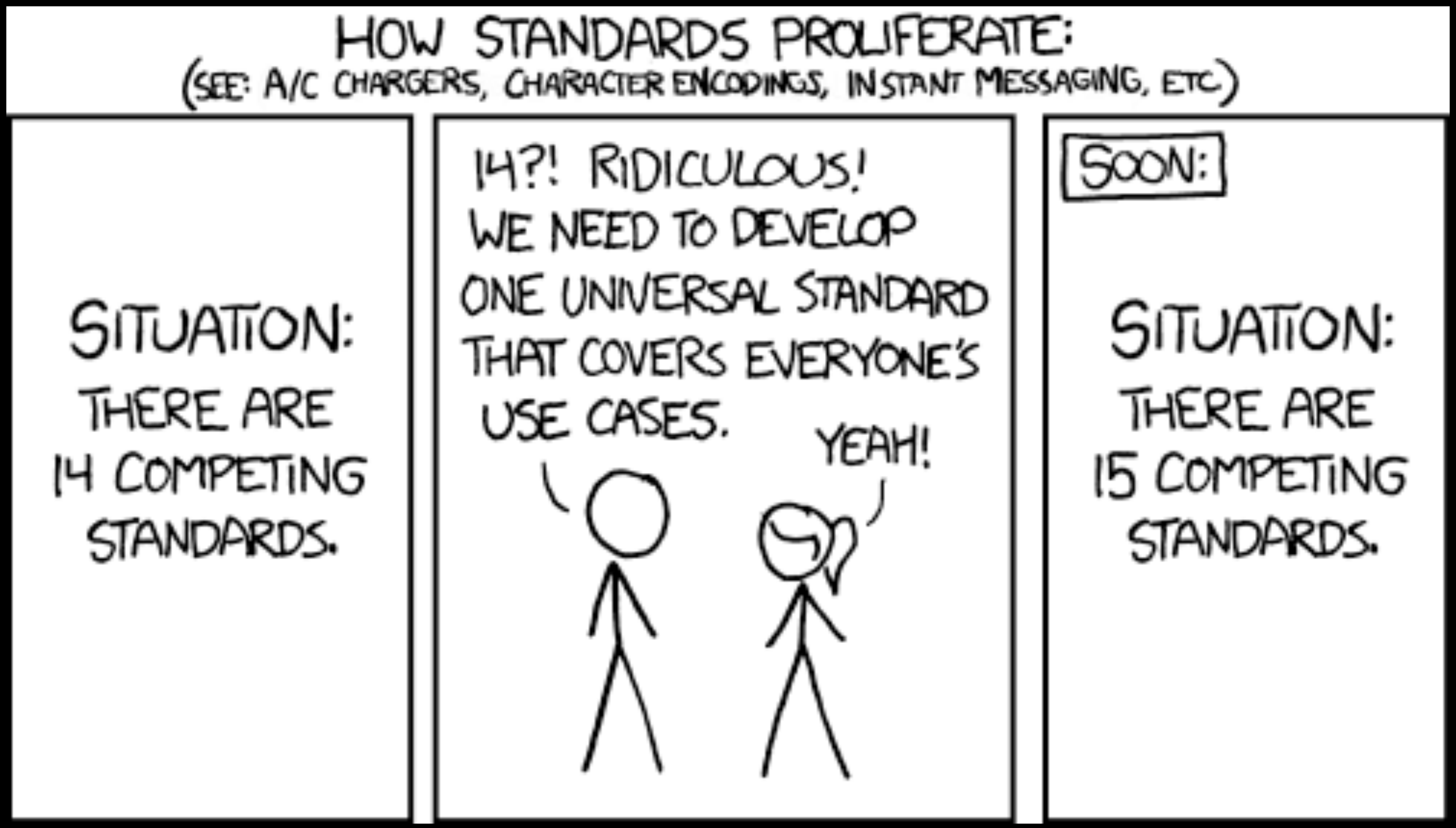
```
whatis("Git is a free and open source distributed version
control system designed to handle everything from small to
very large projects with speed and efficiency. - Homepage:
http://git-scm.com/“)

conflict("git")

prepend_path(“LD_LIBRARY_PATH","/software/git/1.8.2/lib")
prepend_path("LD_LIBRARY_PATH","/software/git/1.8.2/lib64")
prepend_path(“MANPATH","/software/git/1.8.2/share/man")

[…]
```

Environment Modules: Implementations


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

http://xkcd.com/927/

Environment Modules: Implementations

- "Classical Modules" (Furlani et. al.): C/Tcl	modules.sourceforge.net
- still in use at arguably most HPC sites	
- modules-tcl (Kent Mein of UMN): Tcl	http://sourceforge.net/p/modules/modules-tcl
- Cmod (Per Cederqvist et. al.): C	http://www.lysator.liu.se/cmod/
- Lmod (Robert McLay of TACC): Lua	https://www.tacc.utexas.edu/tacc-projects/lmod
- similar systems:	
- SoftEnv (by and in use at Argonne National Lab): Csh	http://www.mcs.anl.gov/hs/software/systems/softenv/softenv-admin.html
- dotkit (by and in use at Lawrence Livermore National Lab): *Shell	https://computing.llnl.gov/?set=jobs&page=dotkit

Environment Modules: Implementations

▪ “Classical Modules” (Furlani et. al.): C/Tcl	modules.sourceforge.net
▪ still in use at arguably most HPC sites	
▪ modules-tcl (Kent Mein of UMN): Tcl	http://sourceforge.net/p/modules/modules-tcl
▪ Cmod (Per Cederqvist et. al.): C	http://www.lysator.liu.se/cmod/
▪ **Lmod (Robert McLay of TACC): Lua**	https://www.tacc.utexas.edu/tacc-projects/lmod
▪ similar systems:	
▪ SoftEnv (by and in use at Argonne National Lab): Csh	http://www.mcs.anl.gov/hs/software/systems/softenv/softenv-admin.html
▪ dotkit (by and in use at Lawrence Livermore National Lab): *Shell	https://computing.llnl.gov/?set=jobs&page=dotkit

Modules in the 21st century: Lmod

- I’ll focus on Lmod from now on, though most things will work with other systems as well
- Very actively maintained by one of the most awesome maintainers we’ve seen in an open-source project
- Active community
- Cool new features:
    - automatic swapping of modules, if applicable
    - conflict detection (i.e. two compilers loaded at the same time)
    - loading of modules with “atleast”, “between” and “latest” w.r.t software versions
    - Spider (searching) and Keywords
    - Caching

Modules in the 21st century: Lmod

- I'll focus on Lmod from now on, though most things will work with other systems as well
- Very actively maintained by one of the most awesome maintainers we've seen in an open-source project
- Active community
- Cool new features:
    - Hooks (e.g. function call every time a module is loaded)
    - tcl2lua wrapper for conversion of old module files
    - shell script/environment to Lua wrapper
    - GPU and Xeon Phi specifics (colors :D)
    - […]

Environment Modules: In Action

- module avail

```
----------------- /net/gmi.oeaw.ac.at/software/mendel/intel-x86_64-sandybridge-avx/modules/devel ------------------
   Autoconf/2.69-goolf-1.4.10                            SCons/2.3.0-ictce-5.3.0-Python-2.7.3          (D)
   Automake/1.13.1-goolf-1.4.10                          SWIG/2.0.4-goolf-1.4.10-Python-2.7.3
   Boost/1.51.0-goolf-1.4.10-Python-2.7.3                ZeroMQ/2.2.0-goolf-1.4.10
   Boost/1.51.0-goolf-1.4.10                      (D)    ant/1.8.4-Java-1.7.0_10
   CMake/2.8.4-goolf-1.4.10                              gperf/3.0.4-goolf-1.4.10
   CMake/2.8.11-goolf-1.4.10                             guile/1.8.8-goolf-1.4.10
   CMake/2.8.12-GCC-4.7.2                         (D)    ncurses/5.9-GCC-4.7.2
   Greenlet/0.4.1-goolf-1.4.10-Python-2.7.3              ncurses/5.9-goalf-1.1.0-no-OFED
   JUnit/4.10-Java-1.7.0_10                              ncurses/5.9-goolf-1.4.10
   LZO/2.06-goolf-1.4.10                                 ncurses/5.9-ictce-5.2.0
   LZO/2.06-ictce-5.3.0                           (D)    ncurses/5.9-ictce-5.3.0                       (D)
   M4/1.4.16-goolf-1.4.10                                pkg-config/0.27.1-goolf-1.4.10
   PCRE/8.12-goolf-1.4.10                                setuptools/0.6c11-goolf-1.4.10-Python-2.7.3
   PCRE/8.12-ictce-5.3.0                          (D)    setuptools/0.7.8-goolf-1.4.10-Python-2.7.3
   PyZMQ/2.2.0.1-goolf-1.4.10-Python-2.7.3-zmq2          setuptools/2.0.1-goolf-1.4.10-Python-2.7.3    (D)
   SCons/2.3.0-goolf-1.4.10-Python-2.7.3
```

(only a small subset of the modules at this site)

Environment Modules: In Action

- module load

```
█ aaron.zauner@login0 [~]> python --version
Python 2.6.8
█ aaron.zauner@login0 [~]> module load Python/2.7.3-ictce-5.3.0
█ aaron.zauner@login0 [~]> python --version
Python 2.7.3
```

Environment Modules: In Action

- module swap

```
█ aaron.zauner@login0 [~]> module swap Python Python/2.7.5-goolf-1.4.10

The following have been reloaded with a version change:
  1) bzip2/1.0.6-ictce-5.3.0 => bzip2/1.0.6-goolf-1.4.10
  2) libreadline/6.2-ictce-5.3.0 => libreadline/6.2-goolf-1.4.10
  3) ncurses/5.9-ictce-5.3.0 => ncurses/5.9-goolf-1.4.10
  4) zlib/1.2.7-ictce-5.3.0 => zlib/1.2.8-goolf-1.4.10

█ aaron.zauner@login0 [~]> python --version
Python 2.7.5
```

Environment Modules: In Action

- module spider

```
aaron.zauner@login0 [~]> module spider setuptools


---------------------------------------------------------------------------------------------------------------
setuptools:
---------------------------------------------------------------------------------------------------------------
   Versions:
      setuptools/0.6c11-goolf-1.4.10-Python-2.7.3
      setuptools/0.7.8-goolf-1.4.10-Python-2.7.3
      setuptools/2.0.1-goolf-1.4.10-Python-2.7.3

---------------------------------------------------------------------------------------------------------------
To find detailed information about setuptools please enter the full name.
For example:

   $ module spider setuptools/2.0.1-goolf-1.4.10-Python-2.7.3
---------------------------------------------------------------------------------------------------------------
```

Environment Modules: In Action

- module list

```
aaron.zauner@login0 [~]> module list

Currently Loaded Modules:
  1) std_env                        12) OpenMPI/1.6.4-GCC-4.7.2
  2) gmi_term_color                 13) hwloc/1.6.2-GCC-4.7.2
  3) gmi_resources                  14) OpenBLAS/0.2.6-gompi-1.4.10-LAPACK-3.4.2
  4) ictce/5.3.0                    15) gompi/1.4.10
  5) icc/2013.3.163                 16) FFTW/3.3.3-gompi-1.4.10
  6) ifort/2013.3.163               17) ScaLAPACK/2.0.2-gompi-1.4.10-OpenBLAS-0.2.6-LAPACK-3.4.2
  7) impi/4.1.0.030                 18) bzip2/1.0.6-goolf-1.4.10
  8) imkl/11.0.3.163                19) zlib/1.2.8-goolf-1.4.10
  9) Python/2.7.5-goolf-1.4.10      20) libreadline/6.2-goolf-1.4.10
 10) goolf/1.4.10                   21) ncurses/5.9-goolf-1.4.10
 11) GCC/4.7.2
```

Environment Modules: In Action

- module display

```
aaron.zauner@login0 [~]> module display GCC/4.7.2

---------------------------------------------------------------------------------------------------------------
   /net/gmi.oeaw.ac.at/software/mendel/intel-x86_64-sandybridge-avx/modules/compiler/GCC/4.7.2:
---------------------------------------------------------------------------------------------------------------
whatis("The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran,     Java, and Ada, as well
as libraries for these languages (libstdc++, libgcj,...). - Homepage: http://gcc.gnu.org/")
conflict("GCC")
prepend_path("CPATH","/net/gmi.oeaw.ac.at/software/mendel/29_04_2013/software/GCC/4.7.2/include")
prepend_path("LD_LIBRARY_PATH","/net/gmi.oeaw.ac.at/software/mendel/29_04_2013/software/GCC/4.7.2/lib")
prepend_path("LD_LIBRARY_PATH","/net/gmi.oeaw.ac.at/software/mendel/29_04_2013/software/GCC/4.7.2/lib64")
prepend_path("LD_LIBRARY_PATH","/net/gmi.oeaw.ac.at/software/mendel/29_04_2013/software/GCC/4.7.2/lib/gcc/x86_64-
unknown-linux-gnu/4.7.2")
prepend_path("MANPATH","/net/gmi.oeaw.ac.at/software/mendel/29_04_2013/software/GCC/4.7.2/share/man")
prepend_path("PATH","/net/gmi.oeaw.ac.at/software/mendel/29_04_2013/software/GCC/4.7.2/bin")
setenv("EBROOTGCC","/net/gmi.oeaw.ac.at/software/mendel/29_04_2013/software/GCC/4.7.2")
setenv("EBVERSIONGCC","4.7.2")
setenv("EBDEVELGCC","/net/gmi.oeaw.ac.at/software/mendel/29_04_2013/software/GCC/4.7.2/easybuild/GCC-4.7.2-
easybuild-devel")
help([[   The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran,
    Java, and Ada, as well as libraries for these languages (libstdc++, libgcj,...). - Homepage: http://gcc.gnu.org/

]])
```

Environment Modules: In Action

..A lot more to show but time is limited..

Modules for FOSS Devs.

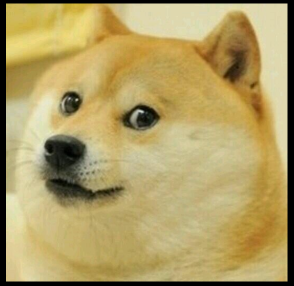
- Still largely unkown and unused in the FOSS community
- Could be a reasonable replacement for the vast number of language specific environment abstraction tools as in use today
- There’s really no reason not to use it (that I know of)
- I’ll demonstrate on two prominent examples

Examples: RVM

- “Ruby enVironment (Version) Manager” [sic!]
- 20k lines of shell script, now being rewritten in Ruby
- basically installs and manages multiple versions of Ruby
- also does a couple of other cool things
- recently did a fundraiser for 50k USD to do the rewrite to Ruby
- actually has corporate sponsoring
- Sorry to say this: I develop Ruby code and It’s a bloated piece of shit that fucks up your environment beyond repair. Won’t ever use it again, ever. ever.
- Of course web developers love the hell out of it

Examples: RVM

- rvm install ruby-2
    - installs Ruby2 from source, downloads dependencies.
    - wow. such magic. many hax. much sophisticate
- rvm list
- rvm use ruby-2
- rvm use 1.9.2 –default
- does any of that sound familiar? right.



Examples: RVM

```
azet@debian:~$ rvm use ruby-2
Using /home/azet/.rvm/gems/ruby-2.1.0
azet@debian:~$ set -v; rvm use system &>gimmethepain
azet@debian:~$ wc -l gimmethepain
6458 gimmethepain
azet@debian:~$
```

Yes. It takes ~6.5k lines of bash code to switch the Ruby version in the current environment with RVM.

"Always code as if the person who ends up maintaining your code is a violent psychopath who knows where you live."

Examples: RVM

- Environmental pollution fact: RVM screws with `cd`.

```
azet@debian:~$ set -x
azet@debian:~$ cd ~
+ cd /home/azet
+ builtin cd /home/azet
+ [[ -n '' ]]
+ true
+ __rvm_cd_functions_set
+ __rvm_do_with_env_before
+ [[ -n '' ]]
+ [[ -n /home/azet/.rvm ]]
+ source /home/azet/.rvm/scripts/initialize
++ : rvm_trace_flag:0
++ ((  rvm_trace_flag > 0  ))
++ [[ -n 4.2.45(1)-release ]]
++ shopt -s extglob
++ export __rvm_env_loaded
++ : __rvm_env_loaded:0:
++ : __rvm_env_loaded:1:
++ [[ -z '' ]]
++ typeset -f __rvm_cleanse_variables
++ __rvm_cleanse_variables
++ __rvm_unset_ruby_variables
++ unset rvm_env_string rvm_ruby_string rvm_ruby_strings rvm_ruby_binary rvm_ruby_gem_home rvm_ruby_gem_path
rvm_ruby_home rvm_ruby_interpreter rvm_ruby_irbrc rvm_ruby_log_path rvm_ruby_major_version rvm_ruby_minor_version
rvm_ruby_package_name rvm_ruby_patch_level rvm_ruby_release_version rvm_ruby_repo_url rvm_ruby_repo_branch
rvm_ruby_revision rvm_ruby_selected_flag rvm_ruby_tag rvm_ruby_version rvm_head_flag rvm_ruby_package_file
rvm_ruby_configure rvm_ruby_name rvm_ruby_url rvm_ruby_global_gems_path rvm_ruby_args rvm_ruby_name rvm_llvm_flag
++ __rvm_load_rvmrc
++ typeset _file
++ typeset -a rvm_rvmrc_files
++ ((  0 == 1  ))
```

- ..this goes on for another 148 lines of executed bullshit. 170 total for every `cd` you type after installing RVM. yup. isn't that great?

Examples: RVM

- Environmental pollution fact: RVM screws with `cd`.

```
azet@debian:~$ set -x
azet@debian:~$ cd ~
+ cd /home/azet
+ builtin cd /home
+ [[ -n '' ]]
+ true
+ __rvm_cd_functio
+ __rvm_do_with_en
+ [[ -n '' ]]
+ [[ -n /home/azet
+ source /home/aze
++ : rvm_trace_fla
++ ((  rvm_trace_f
++ [[ -n 4.2.45(1)
++ shopt -s extglo
++ export __rvm_en
++ : __rvm_env_loa
++ : __rvm_env_loa
++ [[ -z '' ]]
++ typeset -f __rv
++ __rvm_cleanse_v
++ __rvm_unset_rub
++ unset rvm_env_s                                                                   gem_path
rvm_ruby_home rvm_                                                                    minor_version
rvm_ruby_package_n                                                                    branch
rvm_ruby_revision                                                                     e_file
rvm_ruby_configure                                                                    rvm_llvm_flag
++ __rvm_load_rvmr
++ typeset _file
++ typeset -a rvm_
++ ((  0 == 1  ))
```

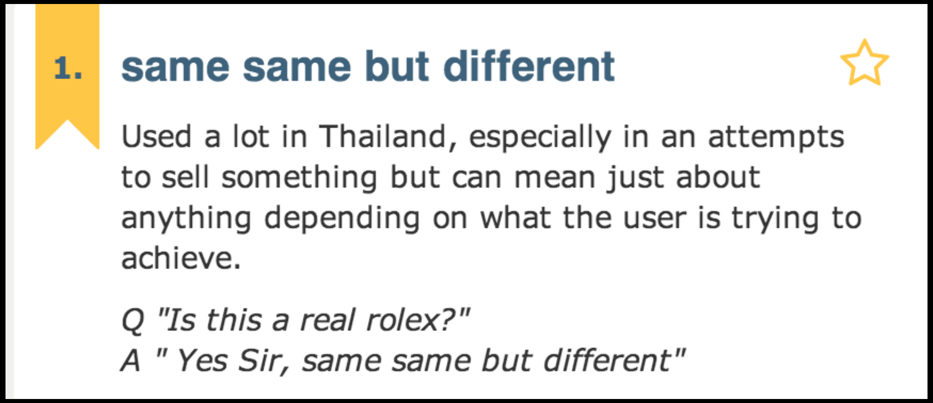
- ..this goes on for another 148 lines of executed bullshit. 170 total for every `cd` you type after installing RVM. yup. isn't that great?

Examples: RVM

```
azet@debian:~$ time cd

real  0m0.009s
user  0m0.004s
sys 0m0.004s
azet@debian:~$ rvm implode
[…]
azet@debian:~$ time cd

real  0m0.000s
user  0m0.000s
sys 0m0.000s
```



1. **same same but different**

Used a lot in Thailand, especially in an attempts to sell something but can mean just about anything depending on what the user is trying to achieve.

Q "Is this a real rolex?"
A " Yes Sir, same same but different"

..if you really do not like modules, please use rbenv instead
a lot more to tell about RVM but lets move on,..

Examples: RVM

Ahh by the way, almost forgot..

```
azet@debian:~$ cd project/
-bash: /home/azet/.rvm/scripts/initialize: No such file or directory
```

Examples: virtualenv

- Basically the same story here
- *"virtualenv is a tool to create isolated Python environments."*

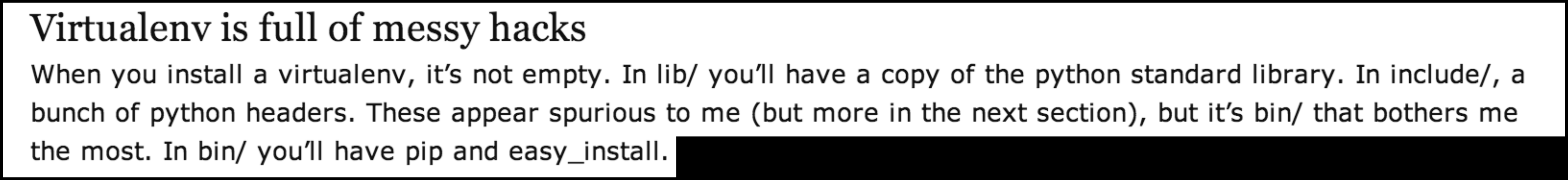
```
azet@debian:~/python$ virtualenv MyLittlePythonSandbox
New python executable in MyLittlePythonSandbox/bin/python
Installing setuptools, pip...done.
azet@debian:~/python$ cd MyLittlePythonSandbox/
azet@debian:~/python/MyLittlePythonSandbox$ source bin/activate
(MyLittlePythonSandbox)azet@debian:~/python/MyLittlePythonSandbox$ pip install XYZFUBAR

[write some code]
```

- People tend to do this a lot and actually ship stuff in that way. So projects end up with this huge requirements.txt files that you might have noticed
- ..if thats not bad development practice I'm not sure what is
- bin/activate is actually a lot cleaner (~15 LoC Bash) than the RVM way

Examples: virtualenv

- It’s a pain installing, using, maintaining and **upgrading** projects that someone has prepared using virtualenv in a production environment
- Often Webprojects use it, which means that your HTTP(S) daemon needs to be aware of virtualenv and switch to the directory and source stuff to get things working.. m(
- From a good recent blog post* on why you should not do it that way:



Virtualenv is full of messy hacks

When you install a virtualenv, it’s not empty. In lib/ you’ll have a copy of the python standard library. In include/, a bunch of python headers. These appear spurious to me (but more in the next section), but it’s bin/ that bothers me the most. In bin/ you’ll have pip and easy_install.

* http://pythonrants.wordpress.com/2013/12/06/why-i-hate-virtualenv-and-pip/

Handling IPhyton with Modules

```
devbox$ module display ipython/0.13.2-Python-2.7.3

whatis("The IPython project provides an enhanced interactive environment that
includes, among other features, support for data visualization and facilities for
distributed and parallel computation. - Homepage: http://ipython.org/index.html”)

conflict("ipython")

load("Python/2.7.3")

prepend_path("LD_LIBRARY_PATH","/software/ipython/0.13.2-Python-2.7.3/lib")
prepend_path("MANPATH","/software/ipython/0.13.2-Python-2.7.3/share/man")
prepend_path(“PATH","/software/ipython/0.13.2-Python-2.7.3/bin")

prepend_path(“PYTHONPATH","/software/ipython/0.13.2-Python-2.7.3/lib/python2.7/
site-packages")

help([[   The IPython project provides an enhanced interactive environment that
includes, among other features, support for data visualization and facilities for
distributed and parallel computation. - Homepage: http://ipython.org/index.html

]])

devbox$ module load ipython/0.13.2-Python-2.7.3
devbox$ ipython —version
0.13.2
devbox$
```

What if I tell you that..

- You can do all this things quite easily with modules
- Save time & effort maintaining stuff
- Have reproducibility
- ..and more fun & time to concentrate on that is important
  - developing good software

?

..but modules does not install stuff for me I take it?

- Back in the 1970ies at Bell Labs Ken Thompson put forward a small set of radical ideas for software engineering that are often ignored by the FOSS community today (who shouted systemd?), those have been summarised by Doug McIlroy of Bell Labs as:

  "Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."

..but modules does not install stuff for me I take it?

- Back in the 1970ies at Bell Labs Ken Thompson put forward a small set of radical ideas for software engineering that are often ignored by the FOSS community today (who shouted systemd?), those have been summarised by Doug McIlroy of Bell Labs as:

  "Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."

- There are a lot of different ways to install binary or source distributions of software.

- They are all compatible with the idea of environment modules.

- For an excellent approach that fits HPC/comp.sci. sites well talk to the EasyBuild guys (the adjacent talk will introduce you to their concept).

- EasyBuild not only does optimised builds but outputs modulefiles as well!

Thank you for your time!

E-Mail: azet@azet.org

XMPP/Jabber: azet@jabber.ccc.de

Twitter: @a_z_e_t

GitHub: azet

GPG Fingerprint: 0xFDB9B5D5