DE LA RECHERCHE À L'INDUSTRIE

Modules

Provides dynamic modification of a user's environment

Xavier Delaruelle <xavier.delaruelle@cea.fr>

FOSDEM 2019
February 3rd 2019, ULB, Bruxelles

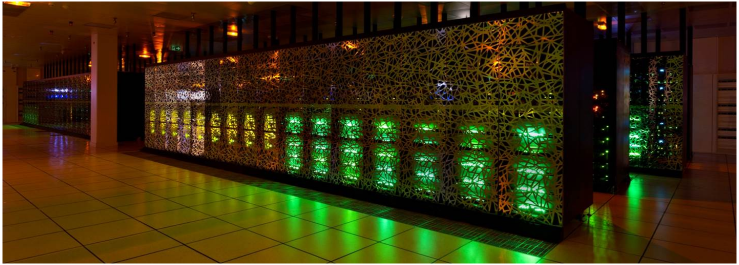
www.cea.fr

What is this about?

- An open source tool that can ease your day-to-day terminal console work
- Project is called *Modules* (or *Environment Modules* for disambiguation)

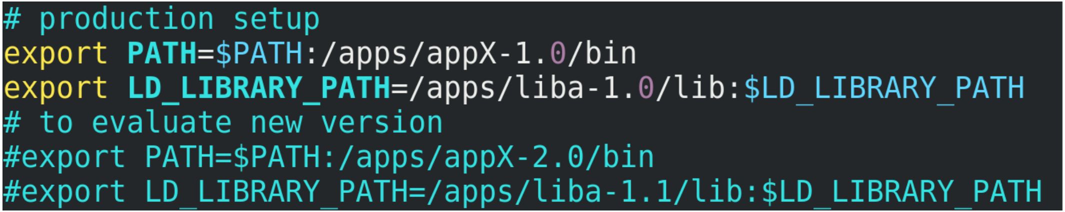
whoami

- I am Xavier Delaruelle
- *Environment Modules* project leader since July 2017
- Work at CEA, a large research institute in France
- In the High Performance Computing (HPC) field



Traditional shell environment configuration

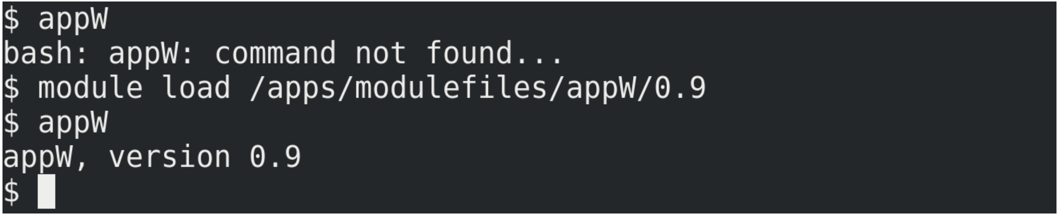
- Everything is put in the shell init file (`.bashrc`, `.profile`, `.zshrc`, `.tcshrc`, ...)
- How to track what have been configured? (hard to distinguish what you have set from the global system setup with `env`/`printenv`)
- How to work with the same user account on multiple projects whose setup are mutually incompatible?



```
# production setup
export PATH=$PATH:/apps/appX-1.0/bin
export LD_LIBRARY_PATH=/apps/liba-1.0/lib:$LD_LIBRARY_PATH
# to evaluate new version
#export PATH=$PATH:/apps/appX-2.0/bin
#export LD_LIBRARY_PATH=/apps/liba-1.1/lib:$LD_LIBRARY_PATH
```

How the Modules project may help?

- It defines a shell function called `module`
- That changes the state of the current shell (environment variables, shell aliases)
- By loading *modulefiles* representing set of environment changes
- Loaded modules are tracked thus they can be unloaded to restore previous environment



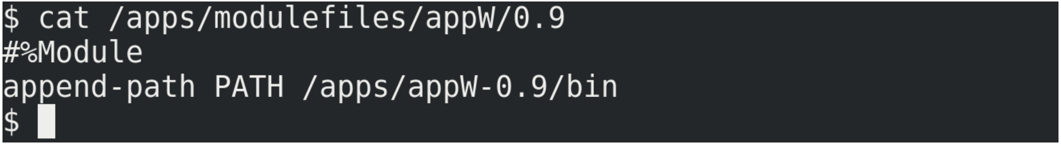
```
$ appW
bash: appW: command not found...
$ module load /apps/modulefiles/appW/0.9
$ appW
appW, version 0.9
$ █
```

How does it work?

The modulefiles

- Modulefiles are scripts describing a set of environment changes
- Written in Tcl + specific environment handling commands:

  `https://modules.readthedocs.io/en/stable/modulefile.html`

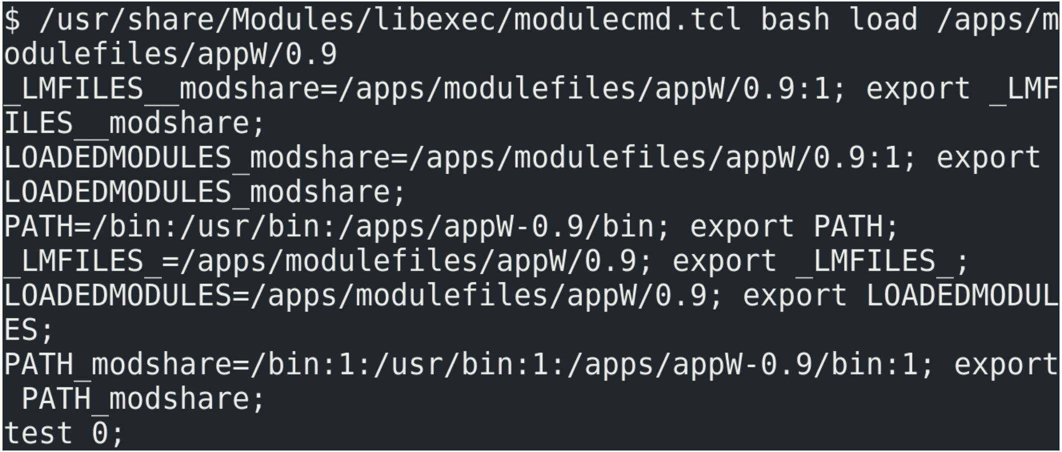


```
$ cat /apps/modulefiles/appW/0.9
#%Module
append-path PATH /apps/appW-0.9/bin
$ █
```

How does it work?

The modulecmd.tcl script

- `modulecmd.tcl` evaluates the sub-commands passed to it to output shell code
- Interprets the modulefiles to produce the shell code to load or unload them

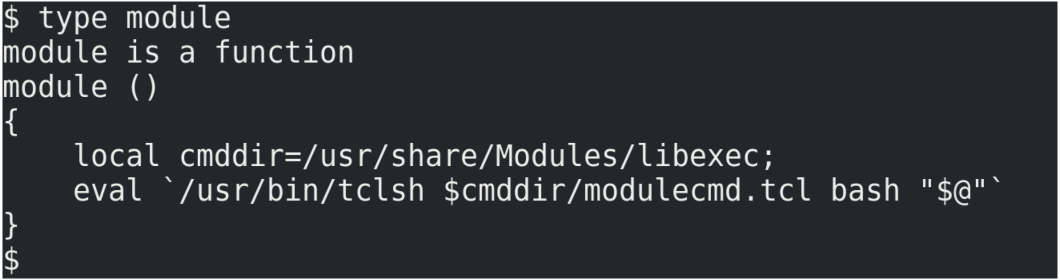


```
$ /usr/share/Modules/libexec/modulecmd.tcl bash load /apps/m
odulefiles/appW/0.9
_LMFILES__modshare=/apps/modulefiles/appW/0.9:1; export _LMF
ILES__modshare;
LOADEDMODULES_modshare=/apps/modulefiles/appW/0.9:1; export
LOADEDMODULES_modshare;
PATH=/bin:/usr/bin:/apps/appW-0.9/bin; export PATH;
_LMFILES_=/apps/modulefiles/appW/0.9; export _LMFILES_;
LOADEDMODULES=/apps/modulefiles/appW/0.9; export LOADEDMODUL
ES;
PATH_modshare=/bin:1:/usr/bin:1:/apps/appW-0.9/bin:1; export
 PATH_modshare;
test 0;
```

How does it work?

The module shell function

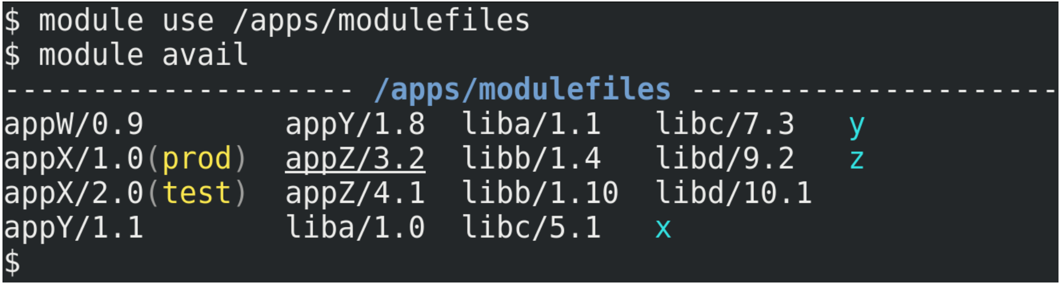
- `module` shell function calls `modulecmd.tcl` script and `eval` its output to update current shell session



```
$ type module
module is a function
module ()
{
    local cmddir=/usr/share/Modules/libexec;
    eval `/usr/bin/tclsh $cmddir/modulecmd.tcl bash "$@"`
}
$
```

Activate catalogs of modulefiles

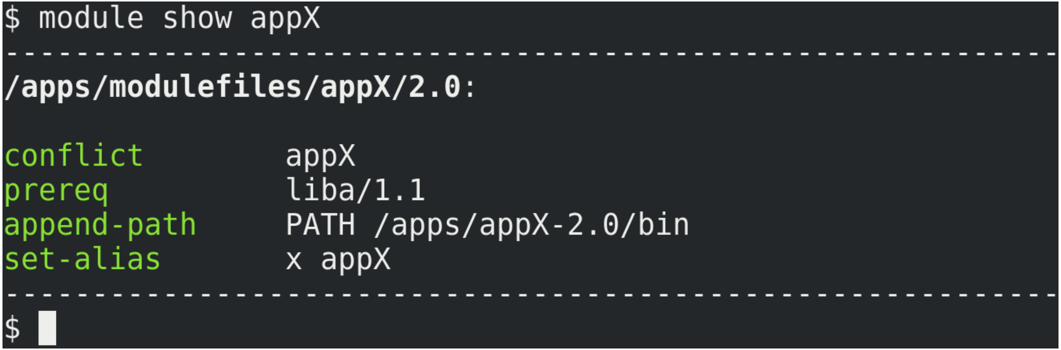
- *Modulepaths* are directories containing modulefiles
- When a modulepath is enabled, `module` search in it to find any modulefiles specified with their short name



```
$ module use /apps/modulefiles
$ module avail
-------------------- /apps/modulefiles ----------------------
appW/0.9        appY/1.8  liba/1.1   libc/7.3   y
appX/1.0(prod)  appZ/3.2  libb/1.4   libd/9.2   z
appX/2.0(test)  appZ/4.1  libb/1.10  libd/10.1
appY/1.1        liba/1.0  libc/5.1   x
$
```

Display content of modulefiles

- Show the modulefile-specific commands written in a given modulefile

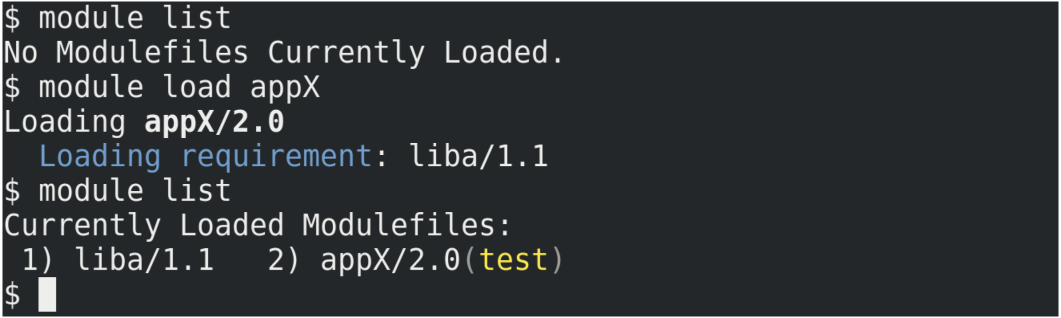


```
$ module show appX
------------------------------------------------------------
/apps/modulefiles/appX/2.0:

conflict        appX
prereq          liba/1.1
append-path     PATH /apps/appX-2.0/bin
set-alias       x appX
------------------------------------------------------------
$ 
```

Complex modulefile load

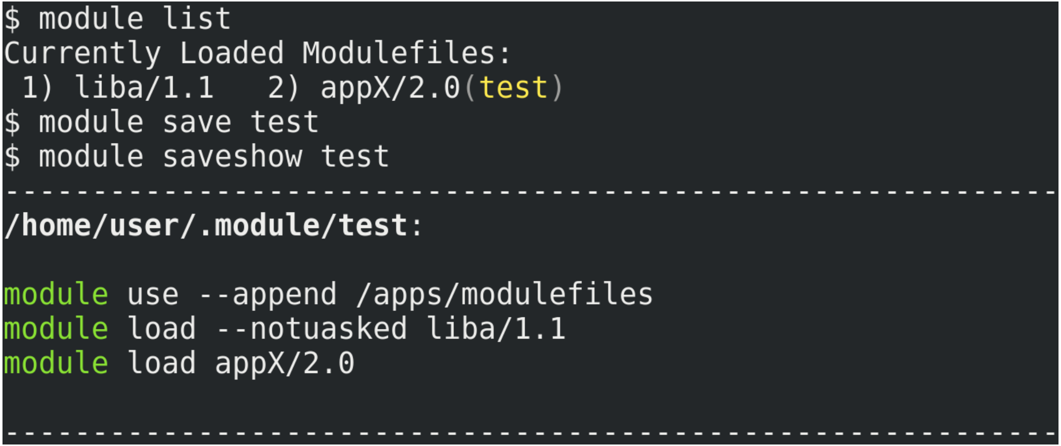
- Resolve dependencies between modulefiles to automatically load or unload them



```
$ module list
No Modulefiles Currently Loaded.
$ module load appX
Loading appX/2.0
  Loading requirement: liba/1.1
$ module list
Currently Loaded Modulefiles:
 1) liba/1.1   2) appX/2.0(test)
$ 
```

Save your current environment in a collection

- Dump current list of enabled modulepaths and loaded modulefiles in a module collection



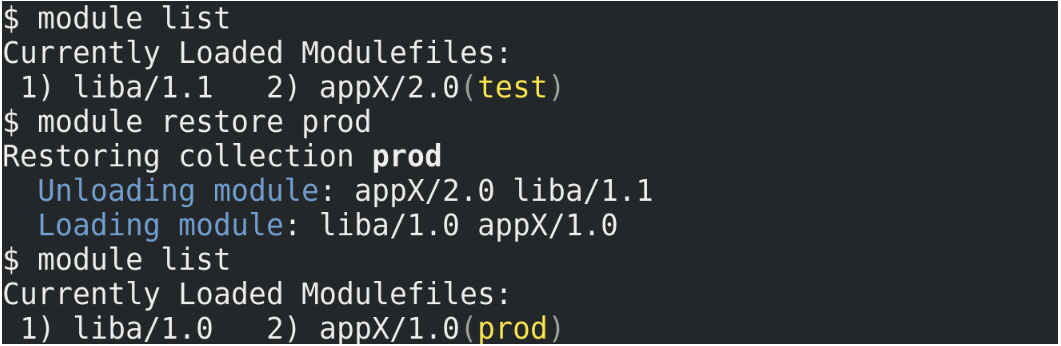
```
$ module list
Currently Loaded Modulefiles:
 1) liba/1.1   2) appX/2.0(test)
$ module save test
$ module saveshow test
------------------------------------------------------------
/home/user/.module/test:

module use --append /apps/modulefiles
module load --notuasked liba/1.1
module load appX/2.0

------------------------------------------------------------
```

Restore a saved environment

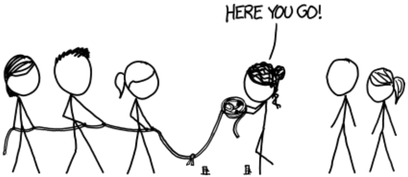
- First, unload enabled modulepaths and modulefiles that are not defined in the collection
- Then, load modulepaths and modulefiles to match the environment state described by collection



```
$ module list
Currently Loaded Modulefiles:
 1) liba/1.1   2) appX/2.0(test)
$ module restore prod
Restoring collection prod
  Unloading module: appX/2.0 liba/1.1
  Loading module: liba/1.0 appX/1.0
$ module list
Currently Loaded Modulefiles:
 1) liba/1.0   2) appX/1.0(prod)
```

As a sysadmin, what Modules could do for you?

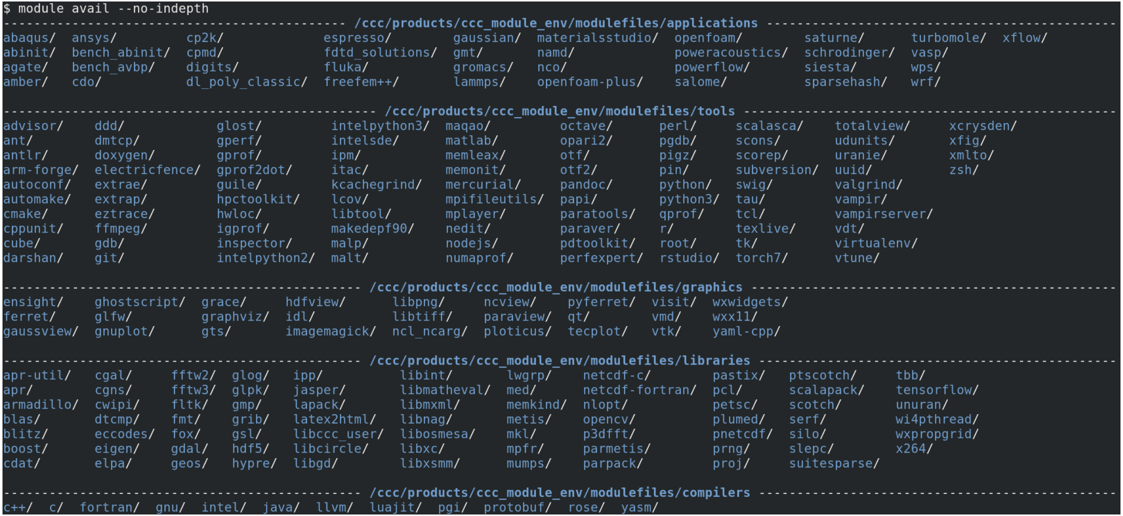
- On shared systems, multiple group of users may have conflicting software needs
- Group 1 wants software a in version 1 whereas Group 2 wants it in version 2
- Cannot used standard installation paths to satisfy everybody



xkcd.com

As a sysadmin, what Modules could do for you?

Give access to complex software catalogue



```
$ module avail --no-indepth
--------------------------------------------- /ccc/products/ccc_module_env/modulefiles/applications ----------------------------------------------
abaqus/  ansys/         cp2k/             espresso/        gaussian/  materialsstudio/  openfoam/        saturne/      turbomole/  xflow/
abinit/  bench_abinit/  cpmd/             fdtd_solutions/  gmt/       namd/             poweracoustics/  schrodinger/  vasp/
agate/   bench_avbp/    digits/           fluka/           gromacs/   nco/              powerflow/       siesta/       wps/
amber/   cdo/           dl_poly_classic/  freefem++/       lammps/    openfoam-plus/    salome/          sparsehash/   wrf/

------------------------------------------------- /ccc/products/ccc_module_env/modulefiles/tools -------------------------------------------------
advisor/    ddd/            glost/         intelpython3/  maqao/         octave/      perl/     scalasca/    totalview/     xcrysden/
ant/        dmtcp/          gperf/         intelsde/      matlab/        opari2/      pgdb/     scons/       udunits/       xfig/
antlr/      doxygen/        gprof/         ipm/           memleax/       otf/         pigz/     scorep/      uranie/        xmlto/
arm-forge/  electricfence/  gprof2dot/     itac/          memonit/       otf2/        pin/      subversion/  uuid/          zsh/
autoconf/   extrae/         guile/         kcachegrind/   mercurial/     pandoc/      python/   swig/        valgrind/
automake/   extrap/         hpctoolkit/    lcov/          mpifileutils/  papi/        python3/  tau/         vampir/
cmake/      eztrace/        hwloc/         libtool/       mplayer/       paratools/   qprof/    tcl/         vampirserver/
cppunit/    ffmpeg/         igprof/        makedepf90/    nedit/         paraver/     r/        texlive/     vdt/
cube/       gdb/            inspector/     malp/          nodejs/        pdtoolkit/   root/     tk/          virtualenv/
darshan/    git/            intelpython2/  malt/          numaprof/      perfexpert/  rstudio/  torch7/      vtune/

----------------------------------------------- /ccc/products/ccc_module_env/modulefiles/graphics ------------------------------------------------
ensight/    ghostscript/  grace/     hdfview/      libpng/     ncview/    pyferret/  visit/  wxwidgets/
ferret/     glfw/         graphviz/  idl/          libtiff/    paraview/  qt/        vmd/    wxx11/
gaussview/  gnuplot/      gts/       imagemagick/  ncl_ncarg/  ploticus/  tecplot/   vtk/    yaml-cpp/

----------------------------------------------- /ccc/products/ccc_module_env/modulefiles/libraries -----------------------------------------------
apr-util/   cgal/     fftw2/  glog/   ipp/          libint/       lwgrp/    netcdf-c/        pastix/   ptscotch/     tbb/
apr/        cgns/     fftw3/  glpk/   jasper/       libmatheval/  med/      netcdf-fortran/  pcl/      scalapack/    tensorflow/
armadillo/  cwipi/    fltk/   gmp/    lapack/       libmxml/      memkind/  nlopt/           petsc/    scotch/       unuran/
blas/       dtcmp/    fmt/    grib/   latex2html/   libnag/       metis/    opencv/          plumed/   serf/         wi4pthread/
blitz/      eccodes/  fox/    gsl/    libccc_user/  libosmesa/    mkl/      p3dfft/          pnetcdf/  silo/         wxpropgrid/
boost/      eigen/    gdal/   hdf5/   libcircle/    libxc/        mpfr/     parmetis/        prng/     slepc/        x264/
cdat/       elpa/     geos/   hypre/  libgd/        libxsmm/      mumps/    parpack/         proj/     suitesparse/

----------------------------------------------- /ccc/products/ccc_module_env/modulefiles/compilers -----------------------------------------------
c++/  c/  fortran/  gnu/  intel/  java/  llvm/  luajit/  pgi/  protobuf/  rose/  yasm/
```

As a sysadmin, what Modules could do for you?

Write shell-agnostic environment guidelines

- Users will also want to use the shell they are used to: `bash`, `ksh`, `tcsh`, `fish`, ...
- Hard to address guidelines to all of them

```
To use the most recent version of GCC:
  BASH/KSH/ZSH: export PATH=$PATH:/apps/gcc/8.2.0/bin
      CSH/TCSH: setenv PATH $PATH:/apps/gcc/8.2.0/bin
          FISH: set -xg PATH $PATH /apps/gcc/8.2.0/bin
```

- With the `module` command, it can be simplified:

```
To use the most recent version of GCC:
  module load gcc/8.2.0
```

Current shell and script language support

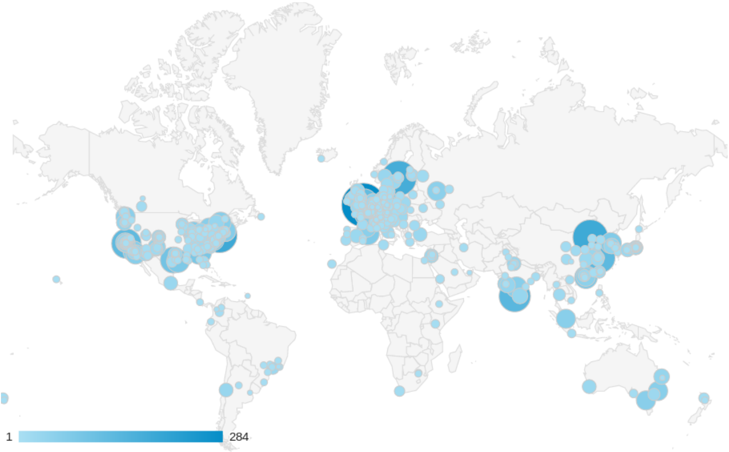
- Most common shells supported:

  `sh · bash · ksh · zsh · csh · tcsh · fish · cmd`

- Also supports scripting languages:

  `tcl · perl · python · ruby · cmake · R`

Where is Modules used?



Modules documentation readers across the world

Project's status

- Environment Modules project has a sustained development pace
- 2 feature releases and multiple bugfix releases per year
- Well integrated in OS repositories
  - RedHat/CentOS/Fedora: `yum install environment-modules`
  - Debian/Ubuntu: `apt-get install modules`
  - openSUSE: `zypper install Modules`
  - Homebrew: `brew install modules`
  - FreeBSD: `pkg install modules`

`https://repology.org/metapackage/environment-modules/versions`

Development trends

- Automatically solve and apply these dependencies when loading or unloading modulefiles
- Implement similar approaches and feature that can be found with package manager tools (like `dnf`, `apt`, etc)

Cool stuff to be developped

- Modulefile cache
- Expiring modulefiles
- Support for modulefiles written in Python
- `module stash` à la `git`, relying on collections

Contributions are welcomed

- Many topics to work on (new shell to support, additionnal modulefile command, support of modulefile written in different languages, <*your idea here*>)
- Heavy non-regression testsuite to guide developpers
  - More than 8000 tests
  - Code largely covered
  - Continuous integration against on multiple Linux distros, OS X, FreeBSD and Windows

build passing | build passing | Ci passing | codecov 99% | docs passing | in repositories 12

Thanks for your attention

- Website: `http://modules.sourceforge.net/`
- Code: `https://github.com/cea-hpc/modules`
- Documentation: `https://modules.readthedocs.io`
- Questions, feedback, new use-cases, want to participate:
  `modules-interest@lists.sourceforge.net`

Commissariat à l'énergie atomique et aux énergies alternatives
Centre de Bruyères-le-Châtel | 91297 Arpajon Cedex
T. +33 (0)1 69 26 40 00 | F. +33 (0)1 69 26 40 00
Établissement public à caractère industriel et commercial
RCS Paris B 775 685 019

DAM
DIF