Modules and Software

Daniel Caunt
Harvard FAS Research Computing

What is Research Computing?

Faculty of Arts and Sciences (FAS) department that handles non-enterprise IT requests from researchers. (*Contact HUIT for most Desktop, Laptop, networking, printing, and email issues.)*

- **RC Primary Services:**
  - Odyssey Supercomputing Environment
  - Lab Storage
  - Instrument Computing Support
  - Hosted Machines (virtual or physical)
- **RC Staff:**
  - 20 staff with backgrounds ranging from systems administration to development-operations to Ph.D. research scientists.
  - Supporting 600 research groups and 3000+ users across FAS, SEAS, HSPH, HBS, GSE.
  - For bio-informatics researchers the Harvard Informatics group is closely tied to RC and is there to support the specific problems for that domain.

FAS Research Computing will be offering a Spring Training series beginning February 2nd. This series will include topics ranging from our Intro to Odyssey training to more advanced job and software topics.

In addition to training sessions, FASRC has a large offering of self-help documentation at https://rc.fas.harvard.edu.

We also hold office hours every Wednesday from 12:00PM-3:00PM at 38 Oxford, Room 206.
https://rc.fas.harvard.edu/office-hours

For other questions or issues, please submit a ticket on the FASRC Portal https://portal.rc.fas.harvard.edu
Or, for shorter questions, chat with us on Odybot
https://odybot.rc.fas.harvard.edu

FAS Research Computing
https://rc.fas.harvard.edu

Intro to Odyssey
Thursday, February 2nd 11:00AM – 12:00PM NWL 426

Intro to Unix
Thursday, February 16th 11:00AM – 12:00PM NWL 426

Extended Unix
Thursday, March 2nd 11:00AM – 12:00PM NWL 426

Modules and Software
Thursday, March 16th 11:00AM – 12:00PM NWL 426

Choosing Resources Wisely
Thursday, March 30th 11:00AM – 12:00PM NWL 426

Troubleshooting Jobs
Thursday, April 6th 11:00AM – 12:00PM NWL 426

Parallel Job Workflows on Odyssey
Thursday, April 20th 11:00AM – 12:00PM NWL 426

Registration not required — limited seating.

HARVARD T.H. CHAN SCHOOL OF PUBLIC HEALTH
Office of Research Strategy and Development

FASRC

ORSD and FASRC will be incorporating a Spring Training series into this semester's Research Computing Office Hours. The trainings will begin in February and will take place during the first 30 minutes of the scheduled office hours so that users may utilize the Office Hours time to ask questions or practice newly acquired or enhanced skills with the support of experts from FASRC.

For additional information about Research Computing at the Harvard Chan School, please visit the ORSD website. Please contact Krista Coleman (kcoleman@hsph.harvard.edu) with questions about the Office Hours or Training Schedule.

For technical questions, please submit a ticket to FASRC or chat with Odybot.

Office Hours
Thursday, January 12th 12:30PM – 2:00PM Kresge 205

Office Hours
Thursday, January 26th 12:30PM – 2:00PM Kresge 205

Intro to Odyssey (first 30 mins of Office Hours)
Thursday, February 9th 12:30PM – 2:00PM Kresge 205

Intro to Unix (first 30 mins of Office Hours)
Thursday, February 23rd 12:30PM – 2:00PM Kresge 205

Extended Unix (first 30 mins of Office Hours)
Thursday, March 9th 12:30PM – 2:00PM Kresge 205

Modules and Software (first 30 mins of Office Hours)
Thursday, March 23rd 12:30PM – 2:00PM Kresge 205

Intro to R
TBA

Choosing Resources Wisely (first 30 mins of Office Hours)
Thursday, April 13th 12:30PM – 2:00PM Kresge 205

Numerical Methods – Part 1
TBA

Troubleshooting Jobs (first 30 mins of Office Hours)
Thursday, April 27th 12:30PM – 2:00PM Kresge 205

Numerical Methods – Part 2
TBA

Parallel Job Workflows (first 30 mins of Office Hours)
Thursday, May 11th 12:30PM – 2:00PM Kresge 205

Objectives

- Feel knowledgeable about computational and software environment
- Understand LMOD software module files
- Know how to handle different types and versions of software applications
- Customize libraries for common scripting languages, such as R, Python and Perl
- Understand basics on version control
- Enable you to “Work smarter, better, faster”

Overview

- Environment basics
- Software module system (LMOD) and software modules
- Installing Java, Python, R and Perl applications
- Installing and updating local packages
- Version control
- Using precompiled software libraries

Environment basics

When you login, Unix executes certain steps for your interactive sessions

- Startup files are read
- Command prompts are set up
- Aliases expanded

Startup files set up default values for your environment

- /etc/profile
- .bash_profile | .bash_login | .profile
- .bashrc

The only things that really need to be in .bash_profile are

- environment variables and
- their exports and commands
- these aren't definitions but actually run or produce output when you log in

Option and alias definitions should go into the environment file .bashrc

Environment basics - .bash_profile

```
[pkrastev@sa01 ~]$ cat .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

export PATH=$PATH:$HOME/bin
```

Environment basics - .bashrc

```
[pkrastev@sa01 ~]$ cat .bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions

alias ls="ls --color=auto"

# LMOD set up
function loadmodules() {
  source new-modules.sh
  module load intel/15.0.0-fasrc01
  module load intel-mkl/11.0.0.079-fasrc02
  module load openmpi/1.8.3-fasrc02
  module load hdf5/1.8.12-fasrc06
  module load matlab/R2015b-fasrc01
  module load totalview/8.8.0.1-fasrc01
}
```

LMOD Module System (1)

LMOD: ENVIRONMENTAL MODULES SYSTEM

https://www.tacc.utexas.edu/research-development/tacc-projects/lmod

Environment Modules provide a convenient way to dynamically change the user's environment through module files (Lua-based scripting files). This includes easily adding or removing directories to the PATH environment variable.

A module-file:

- Contains the necessary information to allow a user to run a particular application or provide access to a particular library. Dynamically changes environment without logging out and back in
- Applications modify the user's path to make access easy
- Library packages provide environment variables that specify where the library and header files can be found

Packages can be loaded and unloaded cleanly through the module system.

- All the popular shells are supported: bash, ksh, csh, tcsh, zsh
- Also available for perl and python
- It is also very easy to switch between different versions of a software package or remove it.

LMOD Module System (2)

Software is loaded incrementally using modules, to set up your shell environment (e.g., PATH, LD_LIBRARY_PATH, and other environment variables)

Using the Harvard-modified, TACC module system LMOD:

- Strongly suggested reading: http://fasrc.us/rclmod

```
source new-modules.sh                      # loads LMOD environment
module load matlab/R2016a-fasrc01          # recommended

module load matlab                         # most recent version
module-query matlab                        # find software modules
module-query matlab/R2016a-fasrc01         # gives more details

module spider matlab                       # finds details on software
module avail 2>&1 | grep -i matlab         # finds titles/defaults
```

So
http

Module loads best placed in SLURM batch scripts:

- Keeps your interactive working environment simple
- Is a record of your research workflow (reproducible research!)
- Keep .bashrc module loads sparse, lest you run into software and library conflicts

Modules: How do they work? (1)

```
[pkrastev@sa01 ~]$ module load gcc/6.1.0-fasrc01
[pkrastev@sa01 ~]$ which gcc
/n/sw/fasrcsw/apps/Core/gcc/6.1.0-fasrc01/bin/gcc
[pkrastev@sa01 ~]$ ll /n/sw/fasrcsw/apps/Core/gcc/6.1.0-fasrc01/
total 2166
drwxr-xr-x 2 root root   1180 Jul  6 17:31 bin
-rw-r--r-- 1 root root 593769 Apr 27 04:20 ChangeLog
-rw-r--r-- 1 root root  18002 Jul 13  2005 COPYING
drwxr-xr-x 3 root root     21 Jul  6 17:29 include
drwxr-xr-x 2 root root    356 Jul  6 17:29 INSTALL
drwxr-xr-x 5 root root   3091 Jul  6 17:30 lib
drwxr-xr-x 6 root root   3551 Jul  6 17:30 lib64
drwxr-xr-x 3 root root     21 Jul  6 17:30 libexec
-rw-r--r-- 1 root root   2625 Jul  6 17:12 modulefile.lua
-rw-r--r-- 1 root root 764169 Apr 27 04:23 NEWS
-rw-r--r-- 1 root root   1026 Jul 16  2012 README
drwxr-xr-x 7 root root    115 Jul  6 17:31 share
```

Modules: How do they work? (2)

```
[pkrastev@sa01 ~]$ cat /n/sw/fasrcsw/modulefiles/Core/gcc/6.1.0-fasrc01.lua
local helpstr = [[
gcc-6.1.0-fasrc01
the GNU Compiler Collection version 6.1.0
]]
help(helpstr,"\n")

whatis("Name: gcc")
whatis("Version: 6.1.0-fasrc01")
whatis("Description: the GNU Compiler Collection version 6.1.0")

---- prerequisite apps (uncomment and tweak if necessary)
for i in string.gmatch("gmp/6.1.1-fasrc02 mpfr/3.1.4-fasrc02 mpc/1.0.3-fasrc04","%S+") do
   if mode()=="load" then
      a = string.match(i,"^[^/]+")
      if not isloaded(a) then
         load(i)
      end
   end
end

---- environment changes (uncomment what is relevant)

setenv("CC" , "gcc")
setenv("CXX", "g++")
setenv("FC" , "gfortran")
setenv("F77", "gfortran")

prepend_path("PATH",              "/n/sw/fasrcsw/apps/Core/gcc/6.1.0-fasrc01/bin")
prepend_path("CPATH",             "/n/sw/fasrcsw/apps/Core/gcc/6.1.0-fasrc01/include")
prepend_path("FPATH",             "/n/sw/fasrcsw/apps/Core/gcc/6.1.0-fasrc01/include")
prepend_path("LD_LIBRARY_PATH",   "/n/sw/fasrcsw/apps/Core/gcc/6.1.0-fasrc01/lib")
prepend_path("LIBRARY_PATH",      "/n/sw/fasrcsw/apps/Core/gcc/6.1.0-fasrc01/lib")
```

Modules: Hierarchies (1)

Use of groupings is important for proper functioning programs.

- Libraries built with one compiler need to be linked with applications with the same compiler version.
- For High Performance Computing there are libraries called **Message Passing Interface (MPI)** that allow for efficient communicating between tasks on a distributed memory computers with many processors.
- Parallel libraries and applications must be built with a matching MPI library and compiler.

Instead of using a flat namespace, we can use module hierarchies.

- Simple technique because once users choses a compiler and MPI implementation, they can only load modules that match that compiler and MPI implementation.
- FASRC follow's TACC's convention:

$MODULEPATH_ROOT/{Core,Comp,MPI} #/n/sw/fasrcsw/modulefiles

Modules: Hierarchies (2)

```
[pkrastev@sa01 ~]$ module-query hdf5

---------------------------------------------------------------------------------------------
  hdf5
---------------------------------------------------------------------------------------------
    Description:
HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed
for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in
their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in
the HDF5 format.  HDF5 is used as a basis for many other file formats, including NetCDF.

    Versions:
      hdf5/1.8.17-fasrc01..... MPI
      hdf5/1.8.16-fasrc03..... MPI
      hdf5/1.8.16-fasrc02..... MPI
      hdf5/1.8.16-fasrc01..... MPI
      hdf5/1.8.15patch1-fasrc01..... MPI
      hdf5/1.8.14-fasrc01..... MPI
      hdf5/1.8.12-fasrc12..... MPI
      hdf5/1.8.12-fasrc08..... Core
      hdf5/1.8.12-fasrc07..... MPI
      hdf5/1.8.12-fasrc06..... MPI
      hdf5/1.8.12-fasrc05..... MPI
      hdf5/1.8.12-fasrc04..... Core
      hdf5/1.8.12-fasrc04..... MPI
      hdf5/1.8.12-fasrc03..... MPI
      hdf5/1.8.12-fasrc02..... MPI
      hdf5/1.8.12-fasrc01..... MPI

    To find detailed information about a module, enter the full name.
    For example,

      module-query hdf5/1.8.12-fasrc01
```

Modules: Hierarchies (3)

```
[pkrastev@sa01 ~]$ module-query hdf5/1.8.16-fasrc03

------------------------------------------------------------------------------------------
  hdf5 : hdf5/1.8.16-fasrc03
------------------------------------------------------------------------------------------
    Description:
        HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of
        datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and
        is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and
        applications for managing, manipulating, viewing, and analyzing data in the HDF5 format.  HDF5 is used as a basis
        for many other file formats, including NetCDF.

    This module an be loaded as follows:
      module load gcc/6.1.0-fasrc01 openmpi/1.10.3-fasrc01 hdf5/1.8.16-fasrc03
      module load gcc/6.1.0-fasrc01 mvapich2/2.2rc1-fasrc01 hdf5/1.8.16-fasrc03
      module load intel/15.0.0-fasrc01 openmpi/1.10.3-fasrc01 hdf5/1.8.16-fasrc03
      module load intel/15.0.0-fasrc01 mvapich2/2.2rc1-fasrc01 hdf5/1.8.16-fasrc03


    This module also loads:
      zlib/1.2.8-fasrc07 szip/2.1-fasrc02
```

Java Programs

- Download the *.jar files or the install files into a home or lab apps/ or bin/ directory
- Include the java CLASSPATH statement in your .bashrc, OR
- Set up a bash environment variable in your .bashrc
- Call the software using the `java` command, pointing to the appropriate routine

```
cd ~
mkdir –p apps; cd apps
wget http://…longURL…/Trimmomatic-0.36.zip
unzip Trimmomatic-0.36.zip
ln –s Trimmomatic-0.36 trimmomatic
echo "export TRIMMOMATIC=$HOME/apps/trimmomatic" >> ~/.bashrc

# in SLURM script or on command line…
module load java/1.8.0_45-fasrc01
cd ~/myFASTQdirectory; mkdir trimmed

# minHeap (-Xms) and maxHeap (-Xmx) options are optional but useful in some cases!!
java -Xms128m –Xmx4g -jar $TRIMMOMATIC/trimmomatic-0.32.jar SE -threads 1 \
  PSG177_TGACCA.fastq.gz trimmed/PSG177_TGACCA.fastq
  ILLUMINACLIP:TruSeq3-PE.fa:2:40:15 LEADING:3 TRAILING:3 \
  SLIDINGWINDOW:4:20 MINLEN:25
```

Python Programs

For Python we recommend:

- Use the standard “module load python/2.7.6-fasrc01” for pulling in default modules
- Use the Anaconda environment for customizing modules & versions
- Multiple custom environments can be set up for home or lab folders (e.g. development or production code). Check conda options for “non-standard” locations
- https://rc.fas.harvard.edu/resources/documentation/software-on-odyssey/python

```
# Load module
module load python/2.7.6-fasrc01
# Create local python environment in ~/.conda/envs/ENV_NAME
conda create -n ENV_NAME --clone="$PYTHON_HOME”
# Use the new environment
source activate ENV_NAME
# Install a new package named MYPACKAGE
conda install MYPACKAGE
# If the package is not available with conda use pip
pip install MYPACKAGE
# If you have problems updating a package first remove it
conda remove PACKAGE
```

R Programs

When loading R from the LMOD software module system, 100s of common packages have already been installed.

Use the R_LIBS_USER environment variable to specify local R package installations:

https://rc.fas.harvard.edu/resources/documentation/software-on-odyssey/r

```
# Load R module, e.g.,
module load R/3.2.0-fasrc01

# Set R_LIBS_USER to your location for R packages, e.g.,
export R_LIBS_USER=$HOME/apps/R:$R_LIBS_USER

# Start R
R

# Inside R, install the desired package, e.g.,
>install.packages("Rcpp")
```

Perl Programs

```
# load Perl, default modules, and set local install
module load perl/5.10.1-fasrc04
module load perl-modules/5.10.1-fasrc11


# can put these in your .bashrc
export LOCALPERL=$HOME/apps/perl          # dir must already exist
export PERL5LIB=$LOCALPERL:$LOCALPERL/lib/perl5:$PERL5LIB
export PERL_MM_OPT="INSTALL_BASE=$LOCALPERL"
export PERL_MB_OPT="--install_base $LOCALPERL"
export PATH="$LOCALPERL/bin:$PATH"

# and now do easy, local installs with cpan, e.g.,
cpan FASTAParse
```

https://rc.fas.harvard.edu/resources/documentation/software-on-odyssey/perl

Using Software Libraries

Libraries allow you to pull in pre-compiled functions and code to your programs
Many are already installed on the cluster, e.g., GSL, BLAS, LAPACK, NetCDF, HDF5, FFTW, MKL, BOOST, and can be loaded as software modules

```
module load  gsl/1.16-fasrc02
```

This will set up environmental variables, such as PATH, LD_LIBRARY_PATH, LIBRARY_PATH, and CPATH

Libraries may also be part of the OS
/lib, /lib64

Linking to specific libraries can be done by setting -l and -L flags, e.g.,

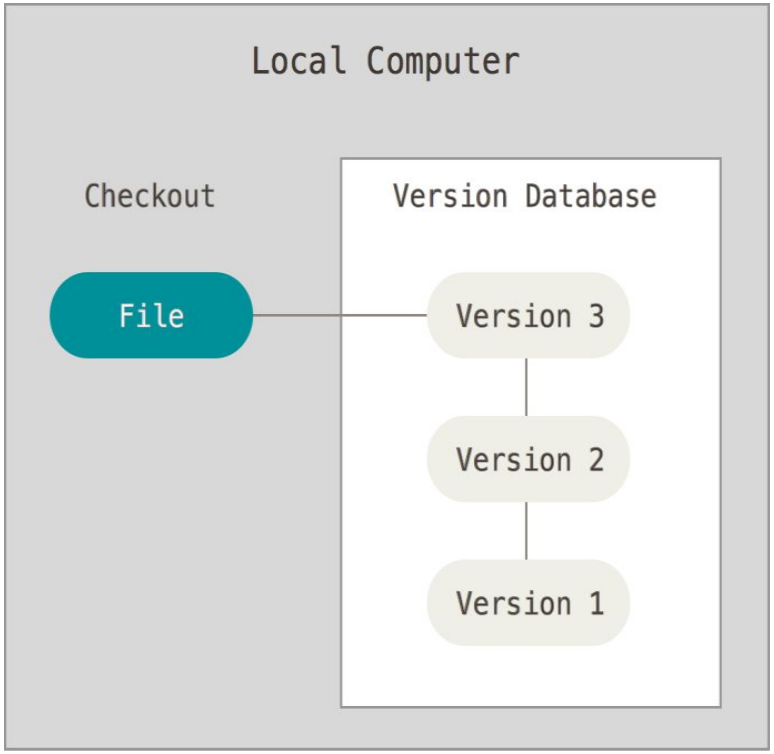
```
gfortran -o my_executable.x my_source.f90 -lblas –llapack

ifort –my_executable.x my_source.f90 -I ${HDF5_INCLUDE} \
            -L ${HDF_LIB} -lhdf5 -lhdf5_fortran
```

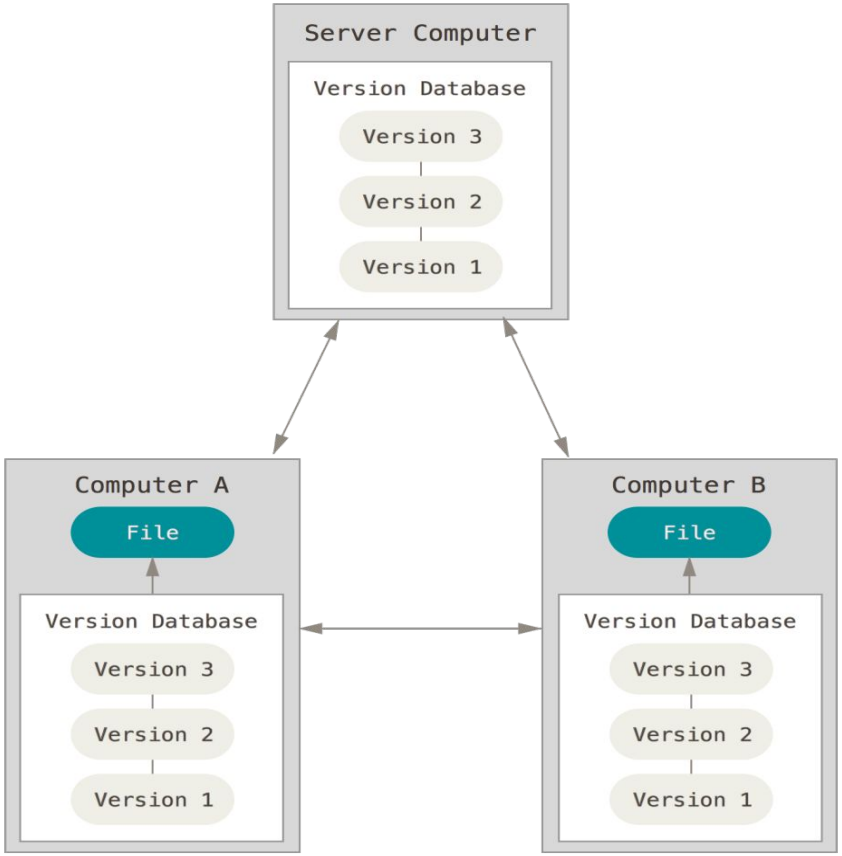
https://github.com/fasrc/User_Codes/tree/master/Libraries

Version Control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

- Typically used for source code files
- In reality you can do this with nearly any type of file on a computer





Request Help - Resources

- https://rc.fas.harvard.edu/resources/support/
  - Documentation
    - https://rc.fas.harvard.edu/resources/documentation/
  - Portal
    - http://portal.rc.fas.harvard.edu/rcrt/submit_ticket
  - Email
    - rchelp@fas.harvard.edu
  - Office Hours
    - Wednesday 12-3pm 38 Oxford - 206
    - @HSPH every other Tuesday 2:00-3:30 pm
  - Training
    - https://www.rc.fas.harvard.edu/training/

Questions ???

Daniel Caunt
Harvard FAS Research Computing