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

Registration not required — limited seating.
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
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-developement/tacc-projects/lmod](https://www.tacc.utexas.edu/research-developement/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/rcmod](http://fasrc.us/rcmod)

```
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
```

Software search capabilities similar to module-query are also available on the RC Portal: [https://portal.rc.fas.harvard.edu/apps/modules](https://portal.rc.fas.harvard.edu/apps/modules)

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

Module: How do they work? (1)

```
[pkrastev@sa01 ~]$ module load gcc/6.1.0-fasrc01
[pkrastev@sa01 ~]$ which gcc
/sw/fasrcsw/apps/Core/gcc/6.1.0-fasrc01/bin/gcc
[pkrastev@sa01 ~]$ ll /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 2 root root 356 Jul  6 17:29 include
  drwxr-xr-x 2 root root 3091 Jul  6 17:30 lib
  drwxr-xr-x 2 root root 3551 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
```
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:

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

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

```
module-query hdf5/1.8.12-fasrc01
```
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

```bash
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
java -Xms128m -Xmx4g -jar $TRIMMOMATIC/trimmomatic-0.32.jar SE -threads 1 \
PSG177_TGACCA.fastq.gz trimmed/PSG177_TGACCA.fastq
```

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

```bash
# 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 \texttt{R_LIBS_USER} environment variable to specify local R package installations:

\url{https://rc.fas.harvard.edu/resources/documentation/software-on-odyssey/r}

```
# Load R module, e.g.,
module load R-packages/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
# dir (must already exist)
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
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
```

\url{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 \texttt{PATH}, \texttt{LD_LIBRARY_PATH}, \texttt{LIBRARY_PATH}, and \texttt{CPATH}.

Libraries may also be part of the OS, 
\texttt{/lib, /lib64}.

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

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

![Diagram of version control system](image-url)
Request Help - Resources

- [https://rc.fas.harvard.edu/resources/support/](https://rc.fas.harvard.edu/resources/support/)
  - Documentation
    - [https://rc.fas.harvard.edu/resources/documentation/](https://rc.fas.harvard.edu/resources/documentation/)
  - Portal
    - [http://portal.rc.fas.harvard.edu/rcrt/submit_ticket](http://portal.rc.fas.harvard.edu/rcrt/submit_ticket)
  - Email
    - [rchelp@fas.harvard.edu](rchelp@fas.harvard.edu)
  - Odybot
    - [https://odybot.rc.fas.harvard.edu/](https://odybot.rc.fas.harvard.edu/)
  - Office Hours
    - Wednesday 12-3pm 38 Oxford - 206
    - @HSPH every other Thursday 12:30-2:00 pm
- Training

Questions ???

Plamen Krastev, PhD
Harvard - Research Computing