Linux Software Installation

Part 2

Qi Sun
Bioinformatics Facility

Two conda distributions

Miniconda:

minimum distribution

Anaconda:

full distribution

Conda channels (repositories)

$HOME/.condarc

bioconda

anaconda

conda-forge

conda install -c bioconda

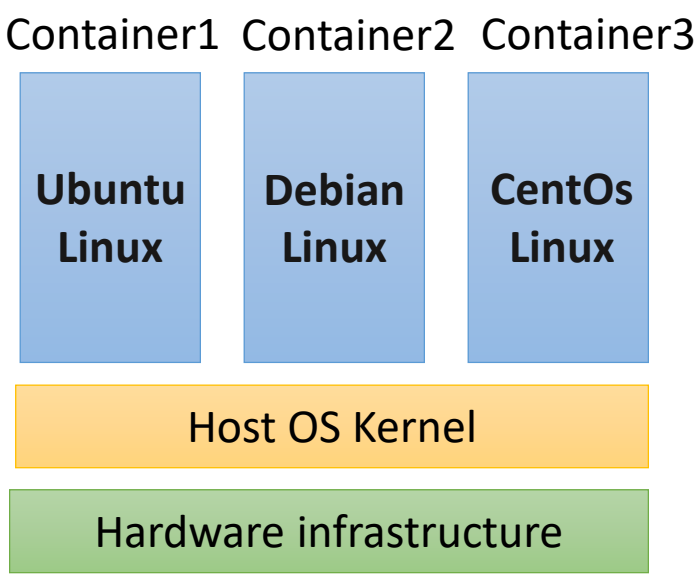
From

Shared libraries

to

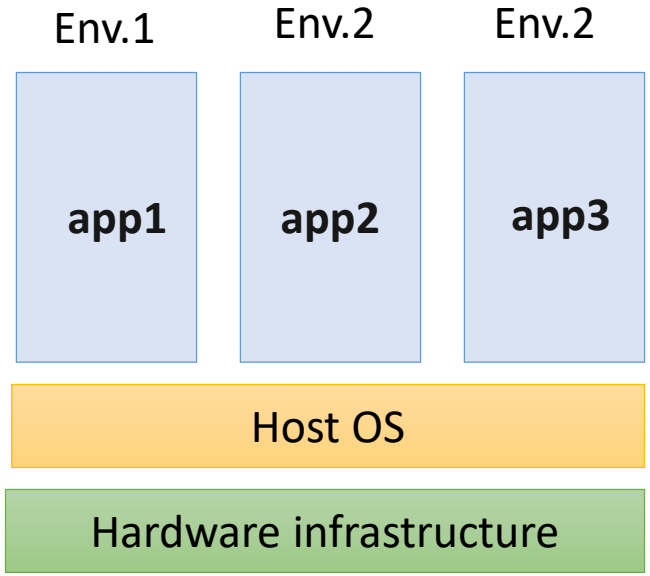
Dedicated libraries for each application

Docker containers



- File system and network port are contained;

Conda environments



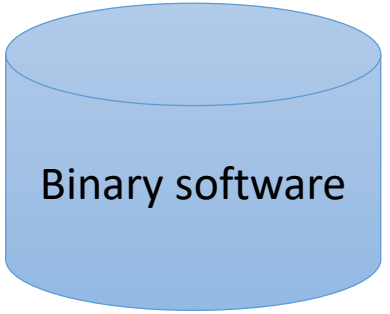
- Modified $PATH;
- File system not isolated;

Binary

Human readable source

Compile

(optimized)



- Hardware & OS dependent

Script

Human readable script

Run

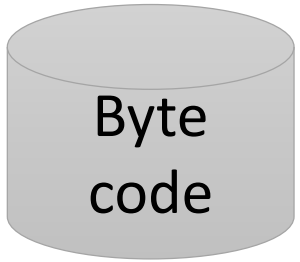
Through Interpreter

- Independent of hardware & OS

Byte code

Human readable source

Compile



Run

Through RTE

(run time environment)

- Independent of hardware & OS

Two modes to run interpreter:

SHELL

```
$ python
>>>import os
>>>os.listdir()
```

Batch script *

```
$ python myscript.py
```

* Alternatively, using shebang line (“#!”)

Read BioHPC software instruction pages:

R: https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=37#c

Rstudio: https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=266#c

Python: https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=556#c

Jupyter: https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=263#c

Conda: https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=574#c

Write permissions when installing software

/usr
/usr/local

Root user has write privilege.

/home/xxxxx
/workdir

You have write privilege.

* When installing software, some modification might be needed to install in $HOME

Python

	PYTHON	PERL	R
Repository	**PYPI ***	**CPAN**	CRAN
Installation tool	**pip**	**cpan**	install.packages
Library PATH	**PYTHONPATH**	**PERL5LIB**	

* pronounced "pie pee eye"

PIP – A tool for installing/managing Python packages

- **PIP use PYPI repository to download software;**
- **Every python command has it companion “pip”.**

On BioHPC:

pip -> python2

pip3 -> python3

On BioHPC

- two versions of Python co-exist

#python 2

pip install myPackage

python myscript.py

#python 3

pip3 install myPackage

python3 myscript.py

In Conda

- Dependent on python version within environment

pip install myPackage

python myscript.py

(either python2 or python3)

Major change for python commands on BioHPC (June 15th,2020)

What stays the same:

python2
pip2

python3
pip3

What is going to change on June 15th:

python
pip

NOW → **2**

June 15th → **3**

Two ways to do “pip install” as a non-root user

pip install deepTools **--user**

Installed in
$HOME/.local/bin
$HOME/.local/lib & lib64

* Suitable for personal installation

pip install deepTools \
**--prefix “mydir" **
[--ignore-installed]

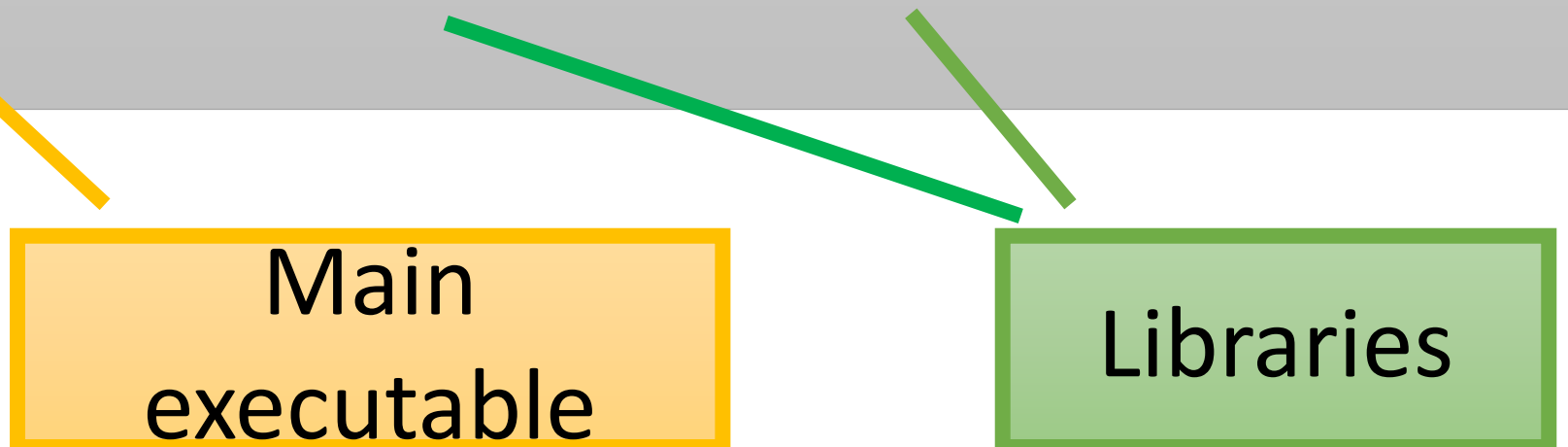
Installed in user defined directory
mydir/bin
mydir/lib & lib64

* Suitable for installation for a group

PIP install all executables and libraries under one directory

```
ls pyGenomeTracks-2.0

bin  lib  lib64
```



PIP parameters to change default behavior

--user

- Install required module in your home directory;
- Skip modules meet requirement;

Parameters to change default behavior

--ignore-installed (-I)

- Install all required modules, present or not;
- Together with **"--prefix=mydir"**

PIP parameters to change default behavior

--upgrade : Upgrade package and all required modules to latest version

PIP would modify Shebang line to the corresponding python:

#!/usr/bin/python2.7

#!/usr/bin/python3.6

```
ls pyGenomeTracks-2.0

bin  lib  lib64
```

To run the software installed by pip:

```
export PATH=/programs/pyGenomeTracks-2.0/bin:$PATH

export PYTHONPATH=/programs/pyGenomeTracks-
2.0/lib64/python2.7/site-packages:/programs/pyGenomeTracks-
2.0/lib/python2.7/site-packages/
```

When using PIP to install, be aware of "which python"

(If not sure, check the shebang line in pip)

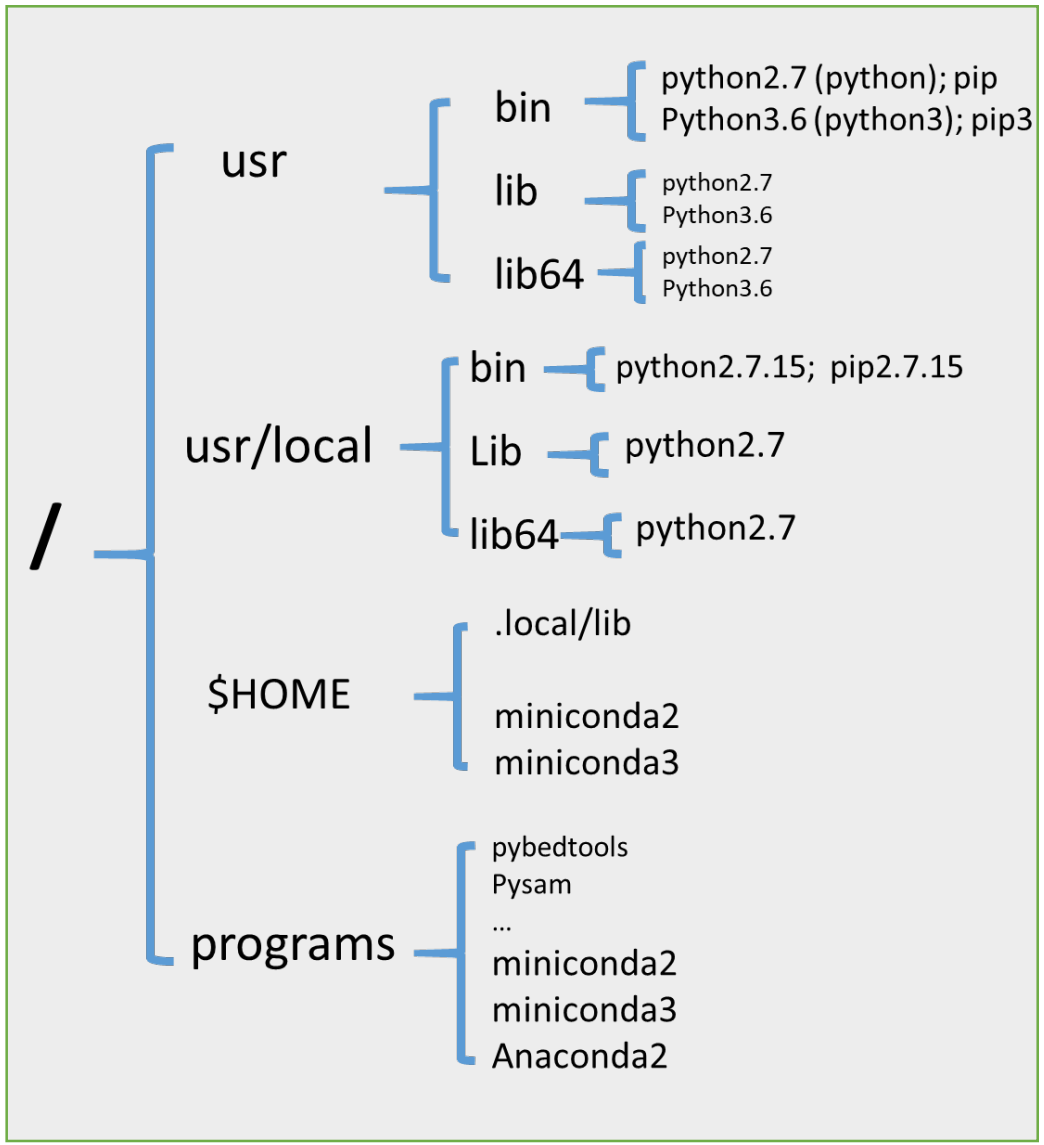
```
which pip
head -n 1 /usr/bin/pip
```

- /usr
  - bin
    - python2.7 (python); pip
    - Python3.6 (python3); pip3
  - lib
    - python2.7
    - Python3.6
  - lib64
    - python2.7
    - Python3.6
- /usr/local
  - bin
    - python2.7.15; pip2.7.15
  - Lib
    - python2.7
  - lib64
    - python2.7
- $HOME
  - .local/lib
  - miniconda2
  - miniconda3
- programs
  - pybedtools
  - Pysam
  - …
  - miniconda2
  - miniconda3
  - Anaconda2

Instructions for Python on BioHPC

https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=556#c

When running PYTHON, be aware of “precedence”



Precedence

Executables

$PATH

- /usr/local/bin
- /usr/bin

Libraries

$PYTHONPATH

- sys.path (installation-dependent default)

- `export PATH=/programs/pybedtools/bin:$PATH`
- `export PYTHONPATH=/programs/pybedbools/lib64`
- `unset PYTHONPATH`

Check which python module is being used

For example:

>>> import numpy

>>> print numpy.__file__
/usr/lib64/python2.7/site-packages/numpy/__init__.pyc

>>> print numpy.__version__
1.14.3

$PYTHONPATH frequently causes problem. E.g. python2 and python3 share the same $PYTHONPATH

* run these commands in “python” prompt

PERL

	PYTHON	PERL	R
Repository	PYPI	**CPAN**	CRAN
Installation tool	pip	**cpan**	install.packages
Library PATH	PYTHONPATH	**PERL5LIB**	

PERL

Check whether a PERL module is present

```
perl -e "use Bio::Seq"
```

or

```
perldoc Bio::Seq
```

Install PERL modules from CPAN

https://cbsu.tc.cornell.edu/lab/doc/Install_PERL_modules.pdf

1. Configure cpan - specify the directory to install;

2. Use cpan to install PERL modules:

```
install XML::Simple
or
force install XML::Simple
```

Configuration of cpan

- Default cpan configuration is ok, and will install PERL modules into your home directory: $HOME/perl

- To reset cpan configuration, you can delete the whole cpan configuration directory: $HOME/.local/share/.cpan

Specify paths of PERL modules installed by you

export PERL5LIB=$HOME/perl/lib/perl5

PERL would search these paths to find a module:

1. **$PERL5LIB defined directories;**

2. **@INC defined directories;**
   Use this command to check:
   perl "-e print @INC“

R

	PYTHON	PERL	R
Repository	PYPI	CPAN	**CRAN**
Installation tool	pip	cpan	**install.packages**
Library PATH	PYTHONPATH	PERL5LIB	

R

Different versions of R on BioHPC

```
ls /programs |grep "^R-"
R-2.13.0
R-2.15.0
R-2.15.2
R-3.0.1
R-3.0.1a
R-3.0.2
R-3.1.0
R-3.2.2
R-3.2.2p
R-3.2.2s
R-3.2.5
R-3.2.5s
R-3.3.2
R-3.3.2s
R-3.4.1
R-3.4.1s
R-3.4.2
R-3.4.2s
```

R Command: R

Default: R-3.5.0

To use another version of R:
export PATH=/programs/R-3.5.0s/bin:$PATH

- This is also applicable to using Rscript command to run R script.

On BioHPC, two separate installations for each R version

R-3.5.0 (Default): parallel BLAS library

R-3.5.0s: regular BLAS library

- Parallel BLAS (Basic Linear Algebra Subprograms) reduces computing time for linear algebra calls by a factor of 3 or more;
- Parallel BLAS could cause '**illegal operand**' errors for some packages;

Install R packages from CRAN

```
# install R package
install.packages("GD")

# load R package
library(GD)
```

* R packages are only installed to the specific R interpreter you are using.

Install R packages from GitHub

```
llibrary(devtools)

install_github("rqtl/qtl2geno")
```

Is something goes wrong when running R, check version and path of R packages (run commands in R shell)

Check version: **packageVersion("edgeR")**

Find package location: **find.package("edgeR")**

Check search path: **.libPaths()**

R studio server

/programs/rstudio_server/rstudio_start **3.6**

- Current supported version: 3.5, 3.5s, 3.5.2, 3.6, 3.6.3, 4.0.0;
- Install R module in the version you want to use;

Issue: C library PATH (LD_LIBRARY_PATH) is not picked up in R studion. Use Docker if needed.

C / C++

Versions of the GCC compiler

Default gcc: 4.8.5

Other versions: 5.5.0, 7.3.0 (/usr/local/gcc-*)

To use a different version of GCC

```
export PATH=/usr/local/gcc-7.3.0/bin:$PATH

export LD_LIBRARY_PATH=/usr/local/gcc-7.3.0/lib:/usr/local/gcc-7.3.0/lib64
```

General installation procedure

1. **Configure**
   - Check CPU, existing libraries, installation targets;
   - Write the compilation plan into the Makefile;

2. **Compile** - from source code to binary code;

3. **Install** - put the executable/library in right locations;

Configure

-Software developers provide instructions how to configure

```
./configure
```

1. **Specify installation directory;**

   To change: ./configure --prefix=/home/xxxxx/bin

2. **Verify the compiler, libraries;**

   Deal with it if libraries missing or not in right version
   (e.g. set LD_LIBRARY_PATH)

Configure – Part 2

Another common way to configure:

cmake

Specify installation directory

```
cmake -DCMAKE_INSTALL_PREFIX:PATH=/home/xxxx/bin
```

Configure – Part 3

Manually edit the “Makefile”

```
PREFIX = /usr/local
LIBDIR = $(PREFIX)/libexec/mafft
BINDIR = $(PREFIX)/bin
MANDIR = $(PREFIX)/share/man/man1
```

Change “PREFIX” to a different directory

Compile

```
make
```

Install

```
make install
```

Run

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

What about JAVA?

Installation: download the **JAR** file

Run software: java -jar myJava.jar

Environment variable for executables path

$PATH

Environment variables for custom library path

PYTHON **$PYTHONPATH**

PERL **$PERL5LIB**

Std binary:$LD_LIBRARY_PATH

Set environment variable:
export PATH=mydirectory:**$**PATH

Check environment variable:
echo **$**PATH