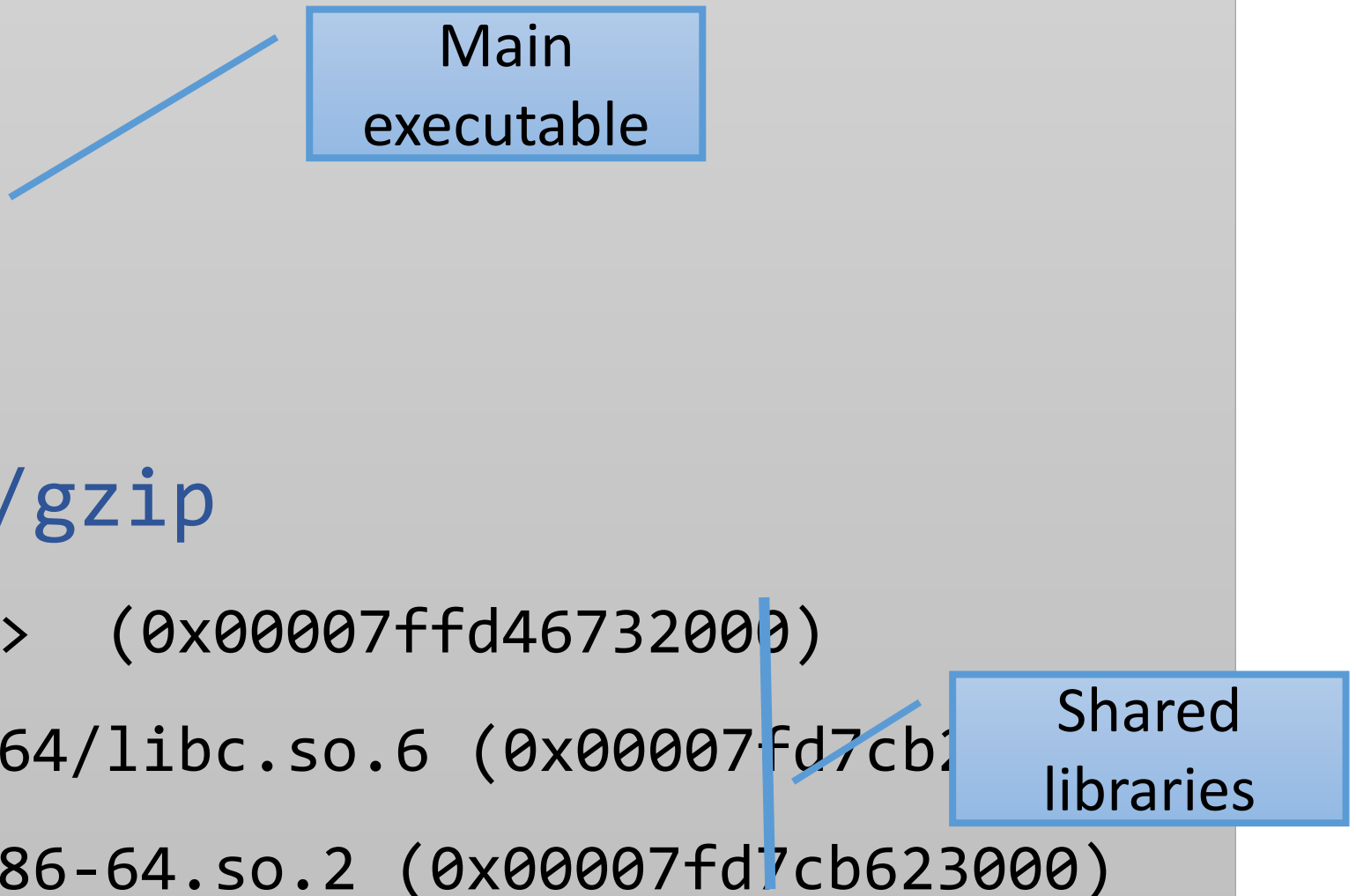
Linux Software Installation

Part 1

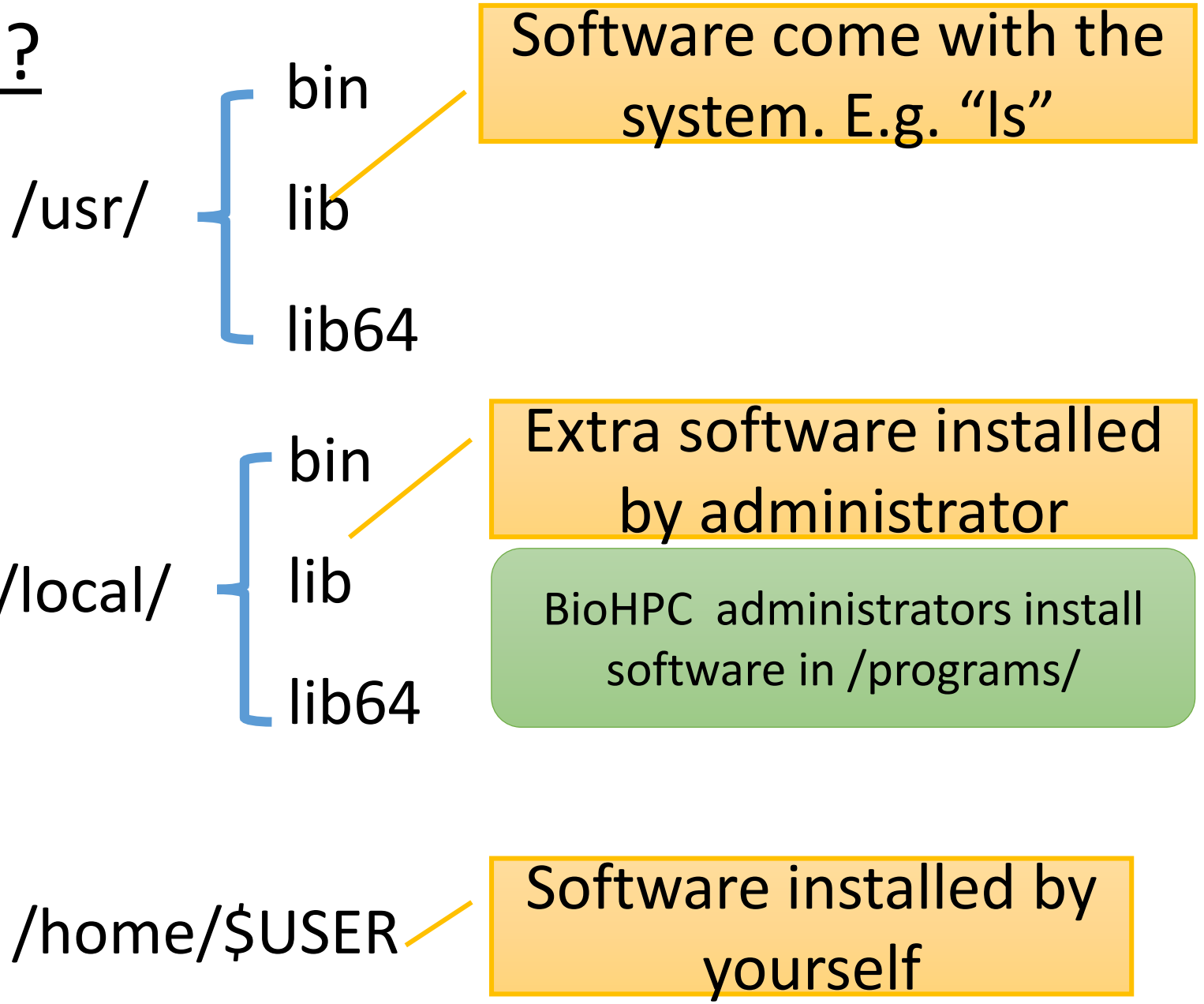
Qi Sun
Bioinformatics Facility

Components of software E.g. gzip

```
$which gzip

/usr/bin/gzip


$ldd /usr/bin/gzip

linux-vdso.so.1 =>  (0x00007ffd46732000)

libc.so.6 => /lib64/libc.so.6 (0x00007fd7cb

/lib64/ld-linux-x86-64.so.2 (0x00007fd7cb623000)
```



Where are the files?



When running a software – 1. executable

The system find the executable file through $PATH

Default $PATH in BioHPC

```
$echo $PATH
/programs/docker/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/programs/bin/mummer:/programs/bin/util:
/programs/bin/bowtie:/programs/bin/bwa:/programs/bin/cufflinks:/programs/bin/samtools:/programs/bin/tophat:/
programs/bin/fastx:/programs/bin/blast:/programs/bin/igv:/programs/bin/velvet:/programs/bin/iAssembler:/progra
ms/bin/GATK:/programs/bin/454:/programs/bin/blat:/programs/bin/perlscripts……
```

Add path to the $PATH variable

```
export PATH=/home/xxxxx/bin:$PATH
```

Use “which” command to find the executable file:

```
which bwa
```

```
/programs/bin/bwa/bwa
```

When running a software – 2. libraries

The system find shared libraries files through /etc/ld.so.conf

Default path for library files

/usr/local/lib64:/usr/local/lib:/usr/lib64:/usr/lib

Add path of extra libraries (dependent on type of software)

Regular	**export LD_LIBRARY_PATH=/home/xxxxx/lib**
PERL	**export PERL5LIB=/home/xxxxx/perl5/5.22.0**
PYTHON	**export PYTHONPATH=/programs/lib/python2.7/site_packages**

Use “ldd” command to identify library files for a compiled binary program

ldd /programs/entropy/bin/entropy

```
linux-vdso.so.1 =>  (0x00007ffefb1d5000)
libgsl.so.0 => /lib64/libgsl.so.0 (0x00007efe3544a000)
libgslcblas.so.0 => /lib64/libgslcblas.so.0 (0x00007efe3520c000)
libz.so.1 => /lib64/libz.so.1 (0x00007efe34ff6000)
libdl.so.2 => /lib64/libdl.so.2 (0x00007efe34df2000)
libm.so.6 => /lib64/libm.so.6 (0x00007efe34aef000)
libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007efe347e7000)
libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007efe345d1000)
libc.so.6 => /lib64/libc.so.6 (0x00007efe3420d000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x00007efe33ff1000)
libsatlas.so.3 => /usr/lib64/atlas/libsatlas.so.3 (0x00007efe333a4000)
/lib64/ld-linux-x86-64.so.2 (0x0000556c6875e000)
libgfortran.so.3 => /lib64/libgfortran.so.3 (0x00007efe33081000)
libquadmath.so.0 => /lib64/libquadmath.so.0 (0x00007efe32e45000)
```

C

System default paths (set by administrator)

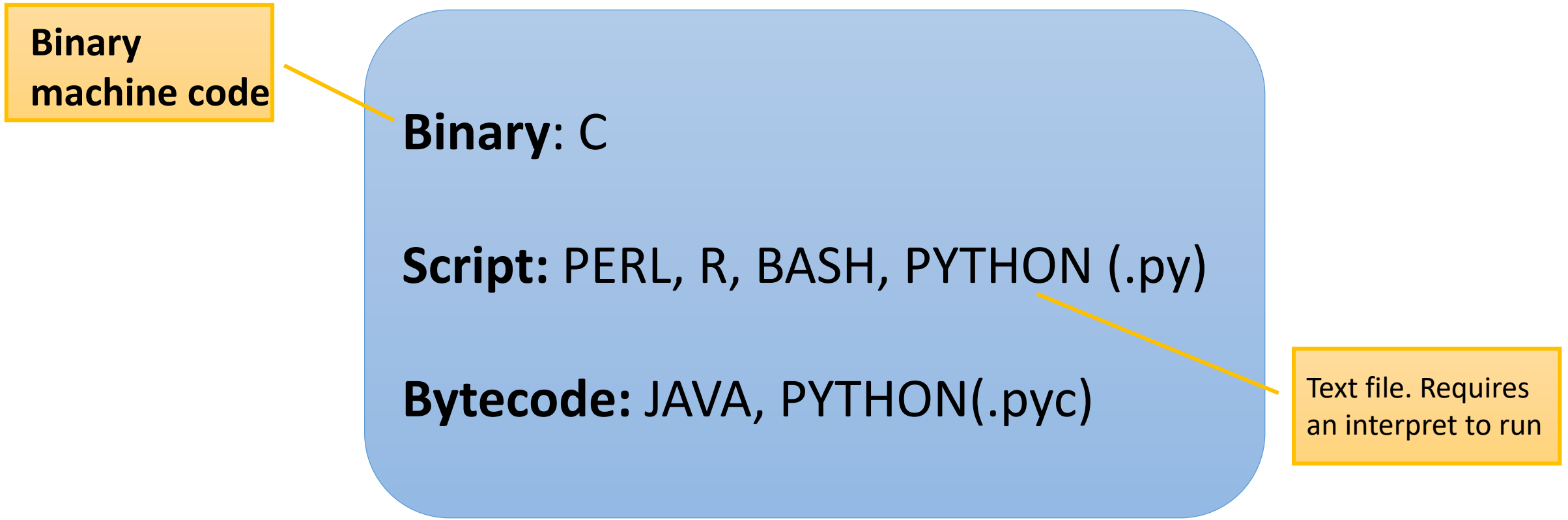
- For executables: $PATH
- For standard libraries: /etc/ld.so.conf

You can add your own paths *:

- For executables: export PATH=<my-sofware>:$PATH
- For std. libraries: export LD_LIBRARY_PATH=<my-library>

* If you add these commands into ~/.bashrc, they would become default for your account

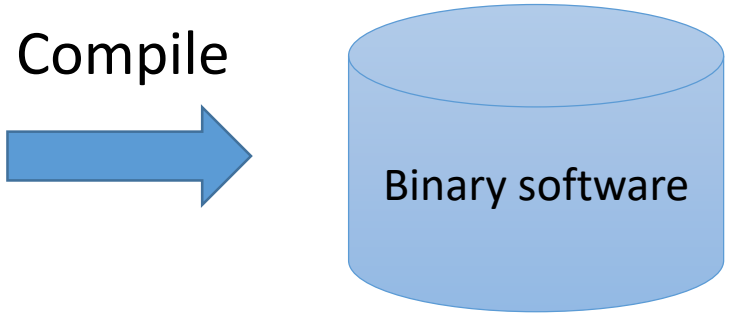
Types of Software: Script vs Binary



Binary

Human readable source code

Compile



- Optimized;
- Hardware & OS restricted

Script

Human readable script

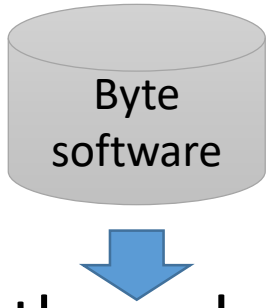
Run through Interpreter

- Not optimized;
- Run across hardware & OS

Byte code

Human readable source code

Compile



Run through runtime environment

- Semi-optimized;
- Run across hardware & OS

A script is a text file, and it requires an interpreter software to run.

Run binary software (does not need another interpreter software)

```
bwa
```

Run script (In the example, it is a python script, and requires python software to interpret)

```
python intron_exon_reads.py
```

You can also run a script without explicitly put interpreter in command

intron_exon_reads.py

Instead of

python **intron_exon_reads.py**

To do this, two requirement:

a. A “shebang” line in the script;
b. File must be executable.

The **Shebang** line is the first line in a script file. It tells Linux system what interpreter to use.

PYTHON

```
#!/usr/bin/python
```

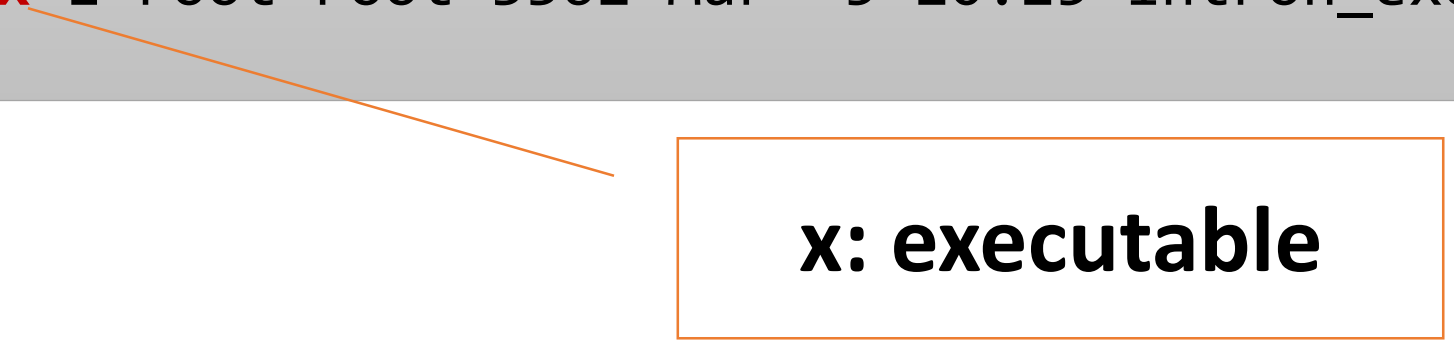
R

```
#!/usr/bin/env Rscript
```

* ***Linux*** is different from ***Windows***. ***Windows*** recognizes the file name extension ".py", and rely on the name extension to decide which interpreter to use.

Make a script file executable:

```
$ls -al intron_exon_reads.py

-rwxr-xr-x 1 root root 3362 Mar  5 20:25 intron_exon_reads.py
```



To make a file executable:

```
$chmod a+x intron_exon_reads.py
```

Software installation - an overview

Static vs Shared Libraries

Static: software with all libraries included

/programs/supernova-2.0.1/supernova

Shared: software require shared libraies

/usr/bin/gzip

/usr/lib64/libc.so.6

/usr/lib64/ld-linux-x86-64.so.2

If available, try to download the “static” version.

You can install multiple versions of the same software

STAR_2.3.0e.Linux_x86_64
STAR_2.4.0d
STAR_2.4.2a
STAR-2.5
STAR-2.5.2b
STAR-2.5.3a

```
export PATH=/programs/STAR-2.5/bin/Linux_x86_64_static:$PATH

which STAR

echo $PATH
```

For software with static libraries, or only using standard system libraries, you can simply download the software, e.g., STAR

wget https://github.com/alexdobin/STAR/raw/master/bin/Linux_x86_64_static/STAR

chmod a+x STAR

Installing software with shared libraries:

- **Versions compatibility**
  - Libraries are shared by multiple software;
  - Different software require different verisons of a library;
- **Path of the libraries**
  - Software cannot find path of library files;
  - The developer’s computer is different from user’s computer.

A version of HiCExplorer requires numpy v1.13

requirements.txt

numpy==1.13.*
scipy==1.0.*
matplotlib==2.1.*
pysam==0.11.*
intervaltree==2.1.*
…

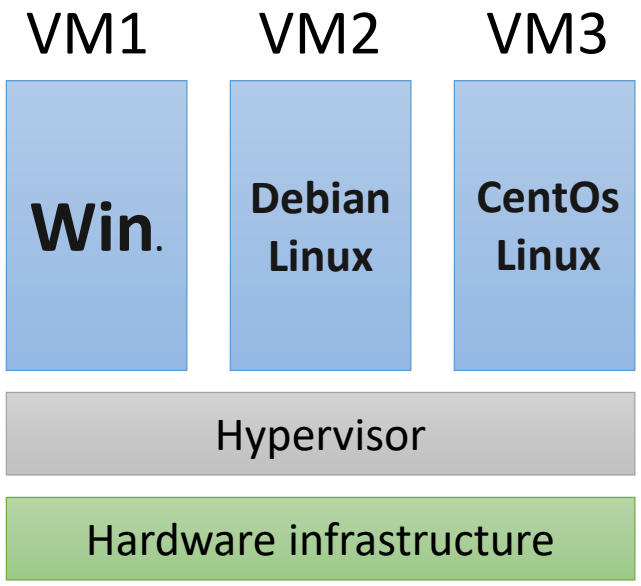
However, our system has version 1.14.3

/usr/lib/python2.7/site-packages/numpy-1.14.3

The solution: Containers and Virtual Environments

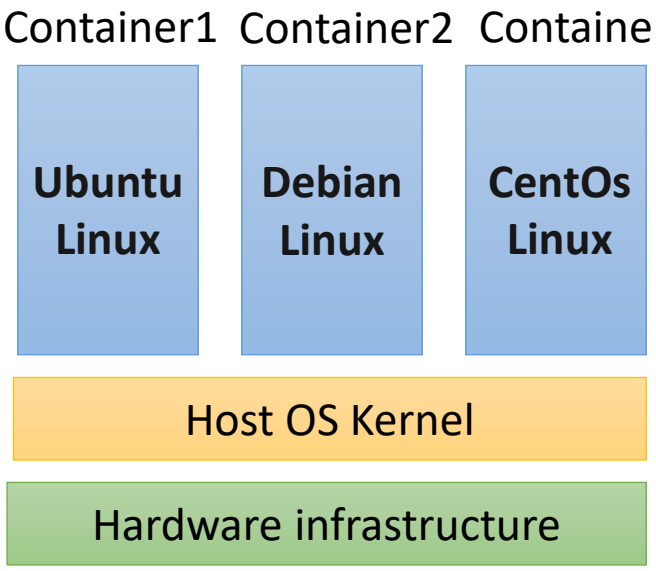
One computer can have many containers/environments. Each software is installed within its own “eco-system”.

Virtual machines



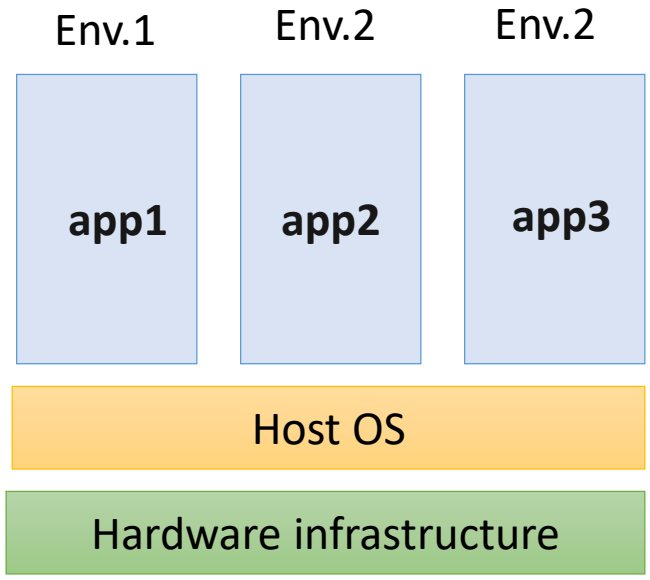
- You can run VM with different OS.

Docker



- File system and network port are contained;

Conda



- Modified $PATH;
- File system not isolated;

File structure of Linux system:

/usr/bin : system executables

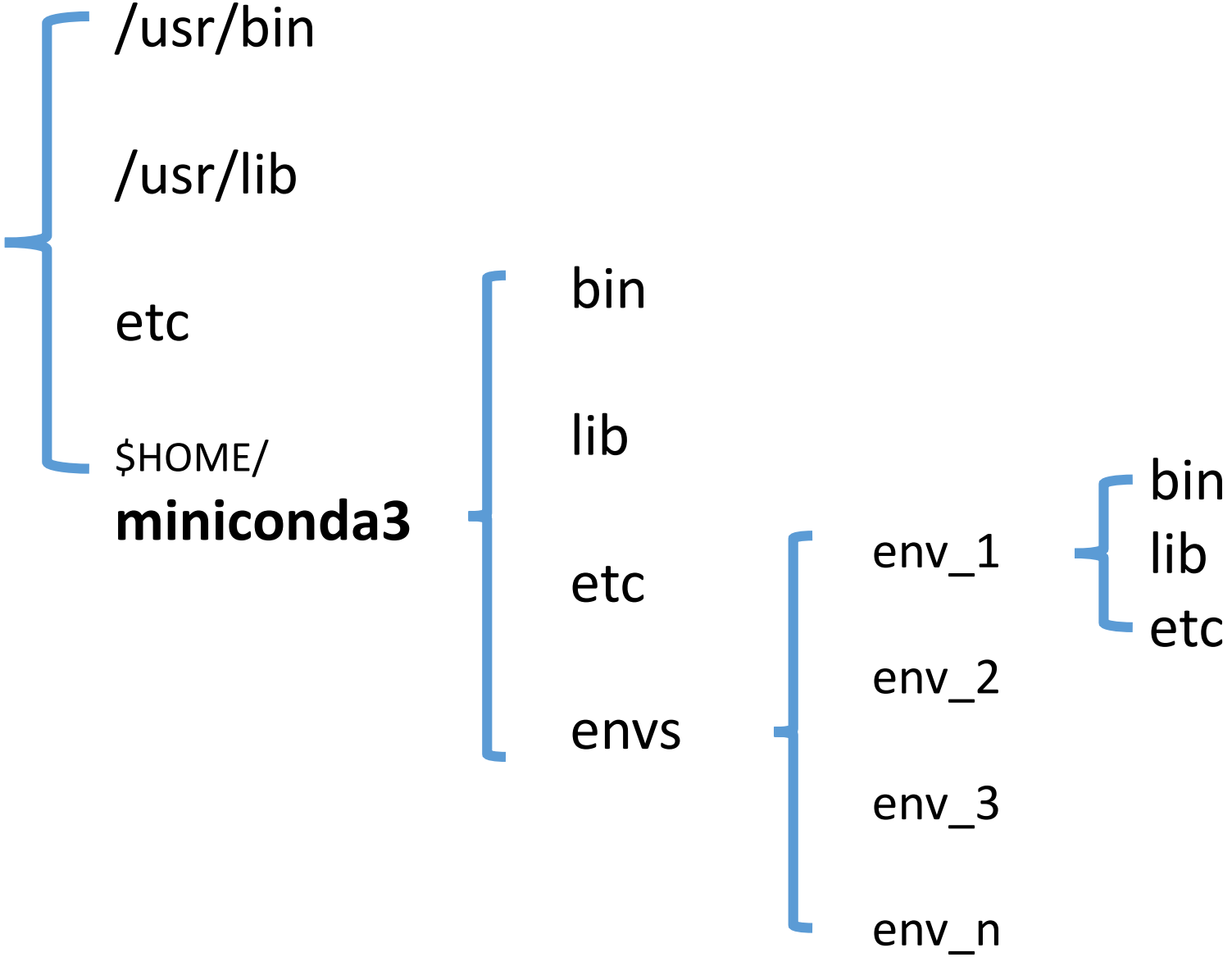
/usr/lib : libraries

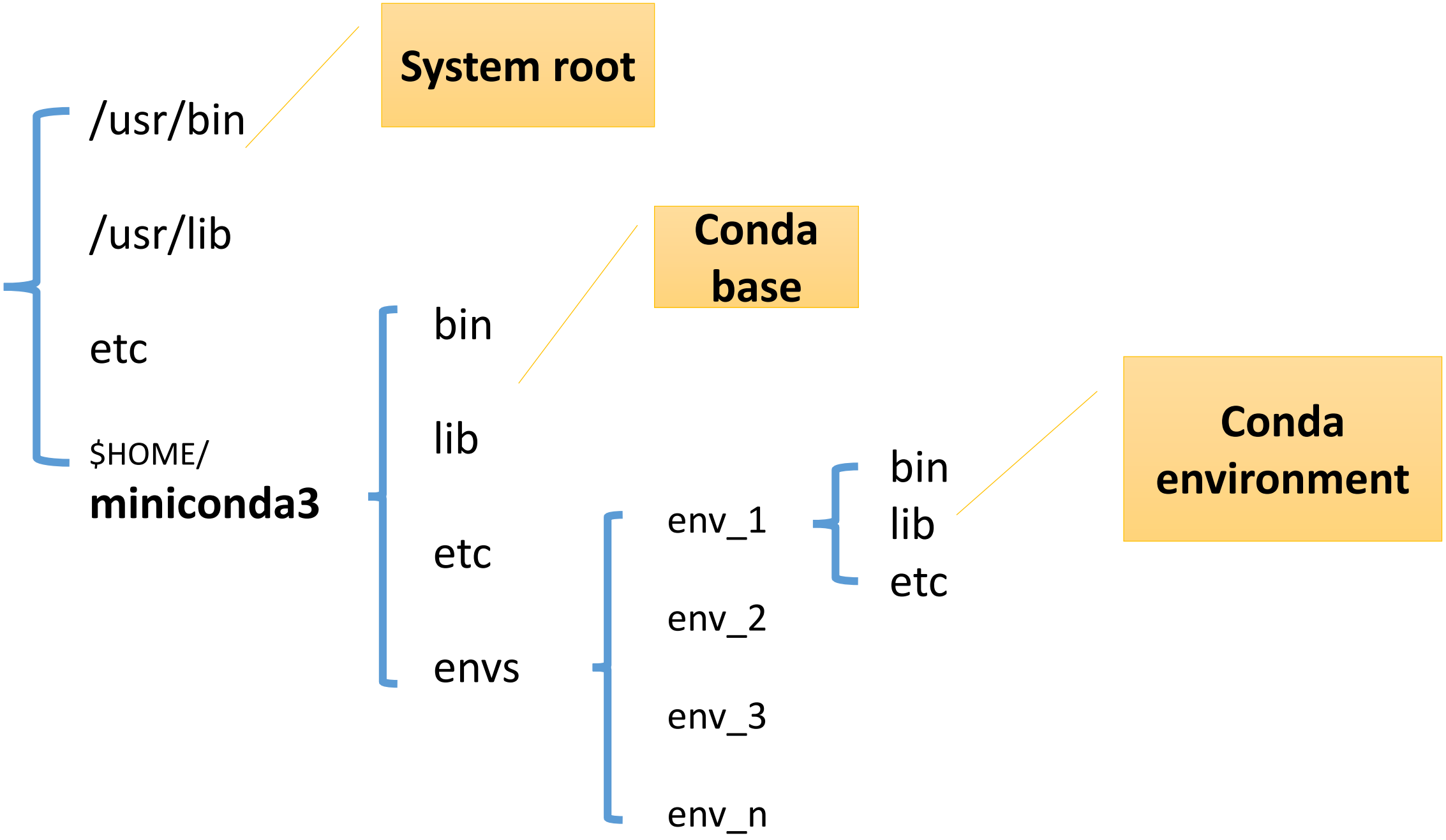
/usr/local/bin : extra executables

/usr/local/lib : extra libraries/modules

/etc : system configuration

With Conda, the file system looks like this:




System root
/usr/bin
/usr/lib
etc
$HOME/
miniconda3
Conda
base
bin
lib
etc
envs
env_1
env_2
env_3
env_n
Conda
environment
bin
lib
etc

System root

Default when login

```
$ which python

/usr/bin/python
```

Conda base

source $HOME/miniconda3/bin/activate

```
(base)$ which python

/home/xxxx/miniconda3/bin/python
```

Conda environment

conda activate env_1

```
(env_1)$ which python

/home/xxxx/miniconda3/env_1/bin/python
```

Anaconda vs Miniconda?

Anaconda for Python2

Miniconda for Python2

Anaconda for Python3

Miniconda for Python3

- Light, no extra libraries;
- Python3 is more used now;

Where to install Conda?

- Default: home directory. (recommended)
- It can also be installed in any directories that you can write to.

How to install Conda?

https://docs.conda.io/en/latest/miniconda.html

Miniconda

	Windows	Mac OS X	Linux
Python 3.7	64-bit (exe installer)	64-bit (bash installer)	64-bit (bash installer)
	32-bit (exe installer)	64-bit (.pkg installer)	32-bit (bash installer)
Python 2.7	64-bit (exe installer)	64-bit (bash installer)	64-bit (bash installer)
	32-bit (exe installer)	64-bit (.pkg installer)	32-bit (bash installer)

Installation instructions

Warning!!! When installing Conda,

you will be prompted this question: **Do you wish the installer to initialize Miniconda3?**

Please answer: **no**

-- Otherwise, all software installed on BioHPC will stop working

How to find out that you have a problem?

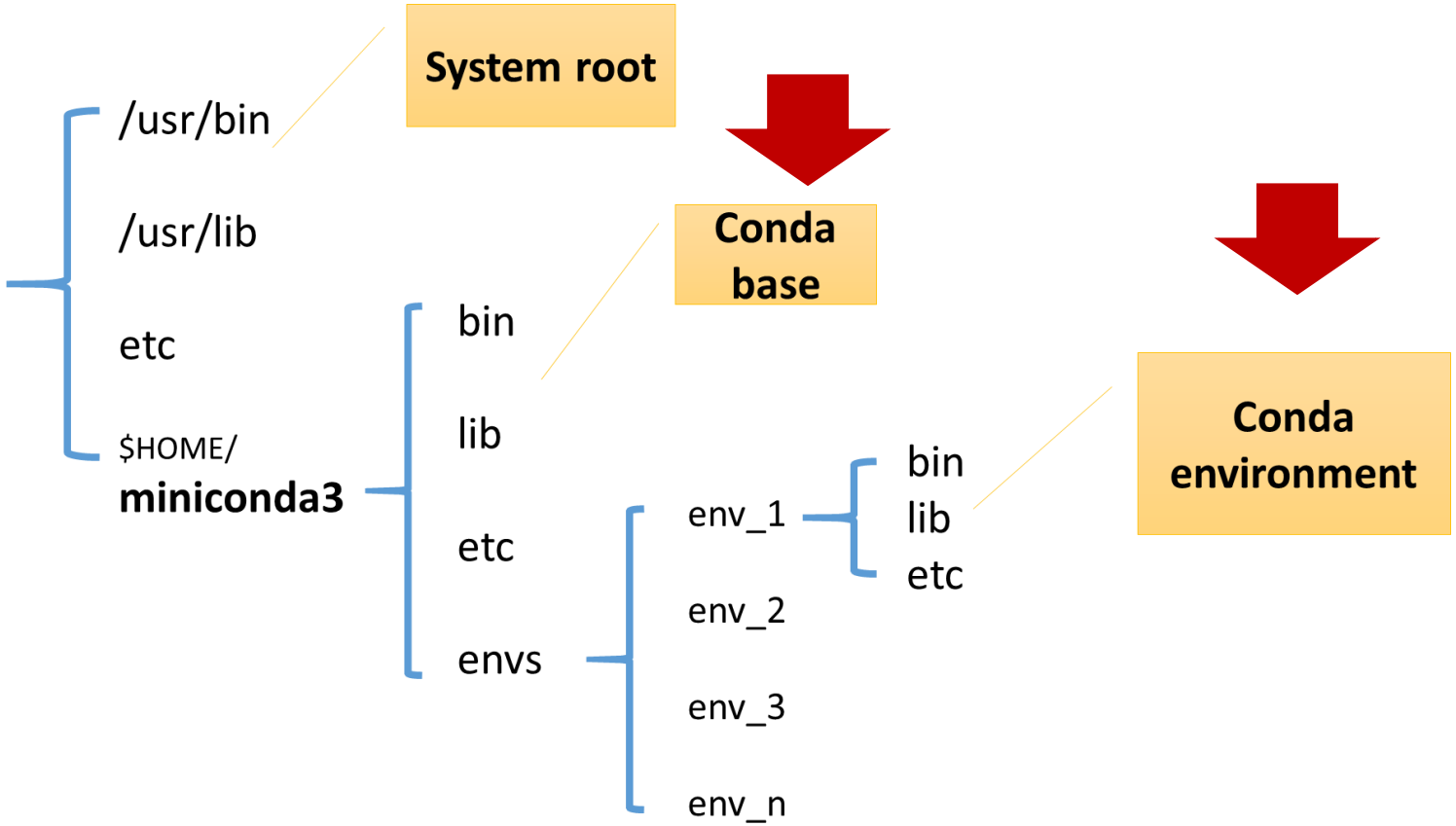
Use the command “which python”, check whether you are using “/usr/bin/python”

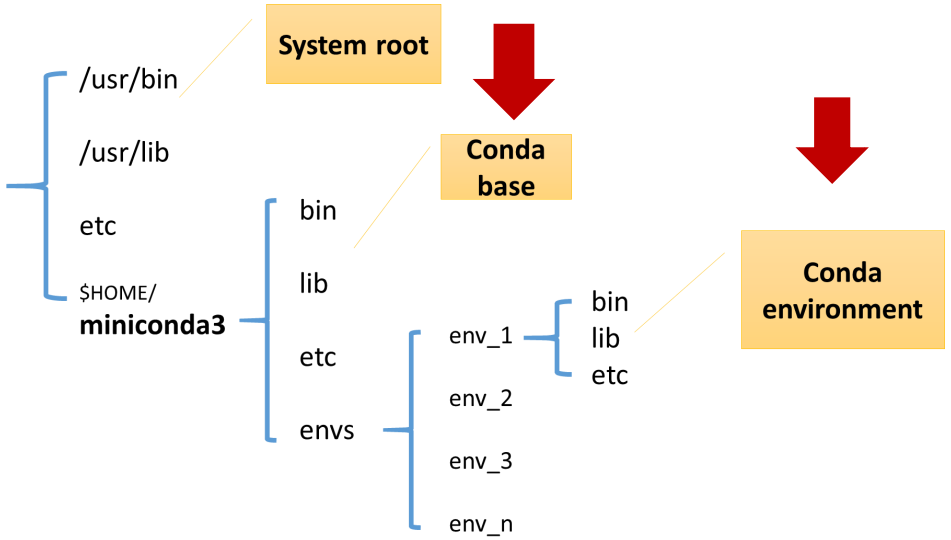
How to correct the problem?

Insert a line “return” before “# >>> conda initialize >>>”

Install software within Conda:

You can install the same software in either of the two levels





Install software in Conda base:

- Save storage space, as the libraries are shared;

Install software in Conda environment:

- No inference between environments;
- Ensure reproducibility

Install software in conda base

```
#start conda
source $HOME/miniconda3/bin/activate
#install software
conda install blast
```

Install software in conda environment

```
#start conda
source $HOME/miniconda3/bin/activate
#install software
conda create -c bioconda -n blast blast
```

Syntax for software installation

```
conda install -c bioconda blast
```

Name of Conda channel. It is the place where conda find the package

```
conda create -c bioconda -n blast blast
```

Name of Conda channel. It is the place where conda find the package

Name of the environment you will create. It can be any name.

Name of the Conda package. This name must exists in the channel.

Run software in conda root

```
#start conda
source $HOME/miniconda3/bin/activate
#run software
blast
```

Run software in conda environment

```
#start conda environment
source $HOME/miniconda3/bin/activate
conda activate blast
#run software
blast
```

Exist conda environment

```
#in a conda environment
conda deactivate

#After this, you are in conda root
```

* There is no simple command to exit conda root to system. You have to exit the Linux session.

Once a conda environment is created, you can install other conda or python modules into the environment

create an empty conda environment with Python v3.6

```
conda create -n myPipeLine python=3.6
```

activate conda environment

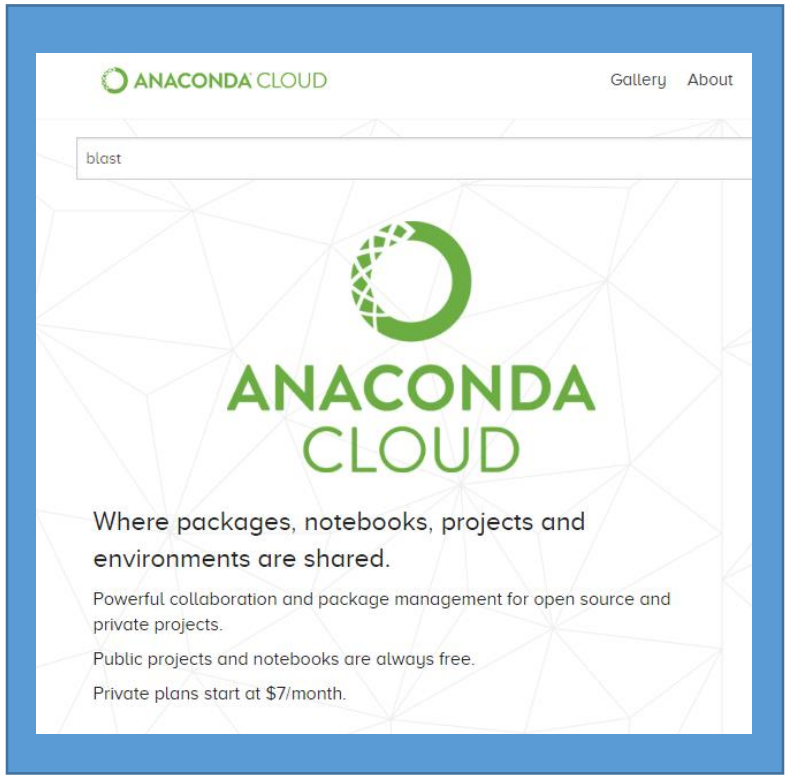
```
conda activate myPipeLine
```

install pysam module in this environment

```
pip install pysam
```

Check the package availability

https://anaconda.org/



Searching for “blast” returns:

Bioconda / blast 2.7.1

BioBuilds / blast 2.6.0

Current version at NCBI: 2.7.1

Check the software version in Conda before you use it

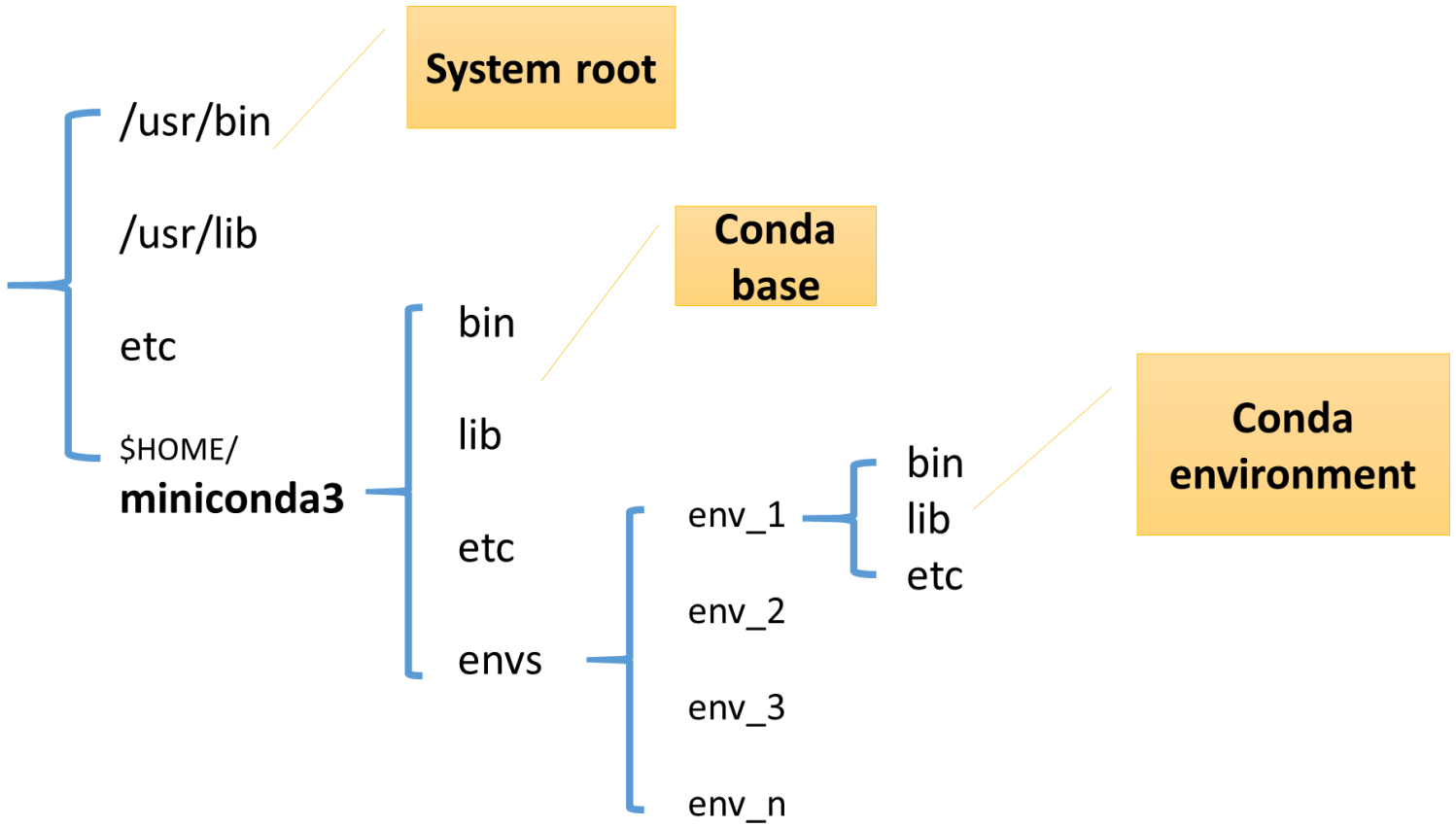
If something goes wrong, you can delete the environment directory or the whole conda directory, and build from scratch

Once you are in Conda, you have full privilege to install anything.

You can install other python modules either with pip or conda to install more python modules. But using “conda install” whenever possible.

```
pip install numpy

conda install numpy
```

When running software in Conda, always make sure that you know what layer you are in. If you are not sure, type “which python”



Trouble shooting when running a software:

- Check which executable you are using. “which python”
- Check which library you are using. Run “echo $PYTHONPATH” for Python or “echo $LD_LIBRARY_PATH” for standard libraries. Use “unset PYTHONPATH” to unset.
- Check your “.bashrc” file, insert a line “return” to avoid problematic code in this file