Introduction to HPC Cluster Computing

Ken-ichi Nomura, Ph.D.

Center for High-Performance Computing
Outline

1. HPC Overview
2. Portable Batch System (PBS)
   - PBS Basics
   - Interactive Mode
   - Job Monitoring
3. Account Management
   - Directory and Quota
   - Computing Time
4. Software Repository
Center for High-Performance Computing

- Foundation for computational research at the University of Southern California.
- World-class academic supercomputer center supported through local investments and without national funding.
- Provides USC faculty and graduate students free access to the facility, HPC tours, trainings and workshops every semester.
- Supports USC research groups in variety of disciplines across campuses.
Computing Services

- Over 2,700 computing nodes (32K CPU cores) on 10G/s Myrinet and 56Gbit/s FDR Infiniband interconnects, 260 GPU (Tesla K20m) nodes
- 2.4 PetaBytes of total storage with GPFS, Panasas, Samfs, NFS
- Over 320 TeraBytes staging storage with OrangeFS
- Cent OS 6.5 Linux, Torque and Moab for resource management and scheduling
- Scientific software and libraries
- Email user support (hpc@usc.edu)
- Online documentations (http://hpcc.usc.edu)
Software Service

- A variety of software from commercial (e.g. MATLAB, Intel & PGI compilers) to open source programs are available
- HPC will assist researchers and install software upon request.
- Researchers are primarily responsible for software & licenses.

<table>
<thead>
<tr>
<th>amber</th>
<th>fftw</th>
<th>cuda</th>
<th>intel</th>
<th>gnu</th>
<th>sas</th>
</tr>
</thead>
<tbody>
<tr>
<td>python</td>
<td>Pegasus</td>
<td>matlab</td>
<td>fdtd</td>
<td>mathematica</td>
<td>iperf</td>
</tr>
<tr>
<td>qespresso</td>
<td>hdf5</td>
<td>globus</td>
<td>gaussian</td>
<td>pgi</td>
<td>spss</td>
</tr>
<tr>
<td>qiime</td>
<td>libroadrunner</td>
<td>hadoop</td>
<td>gromacs</td>
<td>llvm</td>
<td>stata</td>
</tr>
<tr>
<td>R</td>
<td>mpich</td>
<td>hdfview</td>
<td>lammmps</td>
<td>boost</td>
<td>taxila</td>
</tr>
<tr>
<td>git</td>
<td>openmpi</td>
<td>hpctoolkit</td>
<td>NAMD</td>
<td>cellprofiler</td>
<td>bbcp</td>
</tr>
<tr>
<td>opencv</td>
<td>papi</td>
<td>petsc</td>
<td>schrodinger</td>
<td>gurobi</td>
<td>openmpi</td>
</tr>
</tbody>
</table>

USC ITS
Information Technology Services

University of Southern California
User and Job Statistics

<table>
<thead>
<tr>
<th>User Accts</th>
<th>PI Accts</th>
<th>Disk TB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,128</td>
<td>205</td>
<td>1,700</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>General Jobs</th>
<th>Condo Jobs</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>448,864</td>
<td>1,446,750</td>
<td>1,895,614</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Nodes</th>
<th>Condo</th>
<th>General</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10Gb</td>
<td>1728</td>
<td>482</td>
<td>2210</td>
</tr>
<tr>
<td>56.6Gb</td>
<td>143</td>
<td>329</td>
<td>472</td>
</tr>
</tbody>
</table>
HPC trainings and workshops

- Introduction to Linux and the HPC cluster
- Parallel Matlab computing
- GPU and CUDA programming
- Guest lecturing

GPU programming workshop by NVIDIA
Activities

NSF-funded projects:
• Trojan Express Network II (TEN-II) connecting USC campuses with high-speed network
• Advanced Cyberinfrastructure – Research and Education Facilitators (ACI-REFs): Six universities collaborate to empower computational researchers on each campus

USC booth presentation at annual Supercomputing (SC) conference
Outline

1. HPC Overview
2. Portable Batch System (PBS)
   - PBS Basics
   - Interactive Mode
   - Job Monitoring
3. Account Management
   - Directory and Quota
   - Computing Time
4. Software Repository
Portable Batch System (PBS)

• To submit your job to the cluster, create a text file which describes computing resources you need to accomplish your job. This text file is called Portable Batch System (PBS) script.
• Submit the PBS script to job scheduler running on the HPC cluster.
• Your job request will wait in queue until the requested resources become available, the job scheduler will start your job.
PBS Commands: `qsub`  

- `qsub`  
  submit a job to computing cluster

- `-l`  
  resource list

- `nodes`  
  number of nodes

- `ppn`  
  processor per core (nodes attribute)

- `gpu`  
  GPU node request (nodes attribute)

- `nodeset`  
  machine architecture (nodes attribute)

- `mem`  
  amount of total memory

- `pmem`  
  amount of memory per process

- `walltime`  
  wallclock time

- `-d`  
  starting directory

- `-A`  
  specify your account
#PBS script: Example

1: Tell which shell to be used
2: one node with 8 processor per node
3: request for 10 minutes
4: blank
5: comment
6: change directory
7: blank
8: comment
9: Source setup file to use Matlab
10: blank
11: comment
12: run Matlab

```
#!/bin/bash

#PBS -l nodes=1:ppn=8
#PBS -l walltime=00:10:00

cd /home/rcf-proj2/kn1/knomura/

# source setup file (setup.csh for tcsh)
source /usr/usc/matlab/default/setup.sh

# run command
matlab mycode.m
```
PBS script: a bit more advanced

#!/bin/bash
#PBS -l nodes=4:ppn=16:gpu,mem=16GB
#PBS -l walltime=24:00:00
#PBS -A lc_kn
#PBS -d .
#PBS -N my_mpicode

# change to my project directory
cd /home/rcf-proj2/kn1/knomura/

# source necessary setup files for my simulation
source /usr/usc/intel/12.1.1/setup.sh
source /usr/usc/openmpi/1.6.4/share/setup-intel.sh
source /usr/usc/cuda/6.0/setup.sh

# run
mpirun –np 64 my_mpicode > log
Something wrong with this?

```bash
#!/bin/bash

#pbs -l nodes=1,ppn=8
#pbs -l walltime=100:00:00

// change to your project directory
cd /home/rcf-proj2/kn1/knomura/

// source setup file (setup.csh for tcsh)
source /usr/usc/matlab/default/setup.csh

// run command
matlab mycode.m
```
Queues on the HPC cluster

- There are four queues available for public, main, quick, large, and largemem.
- Each queue has different constraints on max. number of queueable jobs, walltime, nodes, simultaneously runnable job.
- The job scheduler automatically select which queue to be assigned on your job depending on the your request. No need to specify queue by users.

<table>
<thead>
<tr>
<th>Queue Name</th>
<th>Maximum Jobs Queued</th>
<th>Maximum Node Count</th>
<th>Maximum Wall Time</th>
<th>Maximum Jobs per User</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>1000</td>
<td>99</td>
<td>24 hours</td>
<td>10</td>
</tr>
<tr>
<td>quick</td>
<td>100</td>
<td>4</td>
<td>1 hour</td>
<td>10</td>
</tr>
<tr>
<td>large</td>
<td>100</td>
<td>256</td>
<td>24 hours</td>
<td>1</td>
</tr>
<tr>
<td>largemem</td>
<td>100</td>
<td>1</td>
<td>336 hours</td>
<td>1</td>
</tr>
</tbody>
</table>

http://hpcc.usc.edu/support/infrastructure/account-resource-limits/
Some Examples:

Q. which queue?

```
#PBS -l nodes=1:ppn=2
#PBS -l walltime=00:59:59

#PBS -l nodes=20:ppn=10,
    walltime=00:59:59,pmem=1gb

#PBS -N myjob
#PBS -d /home/rcf-proj2/kn/knomura
#PBS -l pmem=1gb
#PBS -A lc_kn

#PBS -l nodes=128:ppn=8
#PBS -l walltime=10:00:00
#PBS -N bigjob

#PBS -l nodes=1:ppn=2
#PBS -l walltime=8:00:00
#PBS -l pmem=100gb
#PBS -q largemem

#PBS -l nodes=16:ppn=12
#PBS -l walltime=80:00:00
#PBS -A lc_kn
#PBS -d .
```
## nodes attribute: nodeset

You can specify computer architecture by **nodeset** attribute in case your application needs to run on a certain architecture.

```
[~]$ qsub -l -d . -l nodes=2:ppn=8:pe1950
```

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>#</th>
<th>Node Type</th>
<th>/tmp</th>
<th>Nodeset</th>
</tr>
</thead>
<tbody>
<tr>
<td>hpc0965</td>
<td>hpc0972</td>
<td>8</td>
<td>Dual Hexcore Intel Xeon 3.0 GHz, 24GB</td>
<td>160GB</td>
<td>sl160</td>
</tr>
<tr>
<td>hpc1044</td>
<td>hpc1050</td>
<td>7</td>
<td>Dual Dodecap Core AMD Opteron 2.3 GHz, 48GB</td>
<td>1TB</td>
<td>dl165</td>
</tr>
<tr>
<td>hpc1123</td>
<td>hpc1128</td>
<td>6</td>
<td>Dual Dodecap Core AMD Opteron 2.3 GHz, 48GB</td>
<td>1TB</td>
<td>dl165</td>
</tr>
<tr>
<td>hpc1196</td>
<td>hpc1200</td>
<td>5</td>
<td>Dual Dodecap Core AMD Opteron 2.3 GHz, 48GB</td>
<td>1TB</td>
<td>dl165</td>
</tr>
<tr>
<td>hpc1223</td>
<td>hpc1230</td>
<td>8</td>
<td>Dual Dodecap Core AMD Opteron 2.3 GHz, 48GB</td>
<td>1TB</td>
<td>dl165</td>
</tr>
<tr>
<td>hpc1723</td>
<td>hpc1756</td>
<td>28</td>
<td>Dual Dualcore AMD Opteron 2.3 GHz, 16GB</td>
<td>250GB</td>
<td>x2200</td>
</tr>
<tr>
<td>hpc1872</td>
<td>hpc2081</td>
<td>210</td>
<td>Dual Quadcore Intel Xeon 2.33 GHz, 12GB</td>
<td>60GB</td>
<td>pe1950</td>
</tr>
<tr>
<td>hpc2283</td>
<td>hpc2337</td>
<td>55</td>
<td>Dual Quadcore Intel Xeon 2.5 GHz, 12GB</td>
<td>60GB</td>
<td>pe1950</td>
</tr>
<tr>
<td>hpc2349</td>
<td>hpc2370</td>
<td>21</td>
<td>Dual Quadcore AMD Opteron 2.3 GHz, 16GB</td>
<td>250GB</td>
<td>x2200</td>
</tr>
<tr>
<td>hpc2470</td>
<td>hpc2601</td>
<td>129</td>
<td>Dual Quadcore AMD Opteron 2.3 GHz, 16GB</td>
<td>250GB</td>
<td>x2200</td>
</tr>
<tr>
<td>hpc2758</td>
<td>hpc2761</td>
<td>4</td>
<td>Dual Hexcore Intel Xeon 2.66 GHz, 24GB</td>
<td>120GB</td>
<td>dx360</td>
</tr>
<tr>
<td>hpc3030</td>
<td>hpc3264</td>
<td>236</td>
<td>Dual Octocore Intel Xeon 2.4 GHz, Dual k20 NVIDIA, 64GB</td>
<td>1TB</td>
<td>sl250s</td>
</tr>
<tr>
<td>hpc3386</td>
<td>hpc3389</td>
<td>4</td>
<td>Dual Octocore Intel Xeon 2.4 GHz, 128GB</td>
<td>1TB</td>
<td>sl230s</td>
</tr>
</tbody>
</table>

[http://hpcc.usc.edu/support/infrastructure/node-allocation/](http://hpcc.usc.edu/support/infrastructure/node-allocation/)
nodes attribute: myri and IB

- As mentioned previously, there are two different interconnects in the HPC cluster, called **myrinet** and **Infiniband**.
- On which interconnect your job will run is more important than the architecture since **myrinet** and **Infiniband** are not compatible.
- An executable compiled for myrinet does not run on Infiniband and vice versa.
- Use **myri** and **IB** attribute to specify the interconnect you want to use.

```bash
qusub -l nodes=10:ppn=8:myri,walltime=4:00:00
qusub -l nodes=4:ppn=16:IB,walltime=8:00:00
```
Job Monitoring: `qstat`

- `qstat` show status of PBS jobs
- `-a` all jobs are displayed
- `-u username` display status of specific user’s job
- `-f jobid` display full status of a specific job

[knomura@hpc-login2 ~]$ qstat -u knomura

hpc-pbs.hpcc.usc.edu:

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Username</th>
<th>Queue</th>
<th>Jobname</th>
<th>SessID</th>
<th>NDS</th>
<th>TSK</th>
<th>Memory</th>
<th>Time</th>
<th>S Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>9667416.hpc-pbs. knomura</td>
<td>main</td>
<td>testjob</td>
<td>3827</td>
<td>10</td>
<td>20</td>
<td>--</td>
<td>6:00:00</td>
<td>R</td>
<td>3:21:28</td>
</tr>
</tbody>
</table>
### Job Monitoring: qstat

[~]$ qstat main | head

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Name</th>
<th>User</th>
<th>Time Use S Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>9043999.hpc-pbs</td>
<td>job1</td>
<td>user1</td>
<td>0 Q main</td>
</tr>
<tr>
<td>9447030.hpc-pbs</td>
<td>Job2</td>
<td>user2</td>
<td>0 Q main</td>
</tr>
<tr>
<td>9629959.hpc-pbs</td>
<td>...job.pbs</td>
<td>user3</td>
<td>0 Q main</td>
</tr>
<tr>
<td>9629975.hpc-pbs</td>
<td>...job.pbs</td>
<td>user3</td>
<td>0 Q main</td>
</tr>
<tr>
<td>9633223.hpc-pbs</td>
<td>job3</td>
<td>user4</td>
<td>0 Q main</td>
</tr>
<tr>
<td>9653476.hpc-pbs</td>
<td>...job.pbs</td>
<td>user3</td>
<td>0 Q main</td>
</tr>
<tr>
<td>9676843.hpc-pbs</td>
<td>test.pbs</td>
<td>user5</td>
<td>169:54:2 R main</td>
</tr>
<tr>
<td>9679200.hpc-pbs</td>
<td>rsync</td>
<td>user6</td>
<td>10:17:15 R main</td>
</tr>
</tbody>
</table>
## Job Monitoring: myqueue

**myqueue**  Display jobs status and allocated node list for running jobs.

```bash
[~]$ myqueue
hpc-pbs.hpcc.usc.edu:
Req'd    Req'd    Elap    Job ID   Username   Queue   Jobname   SessID
NDS   TSK   Memory   Time   S Time
---------------------------------------------------------------
7456587.hpc-pbs.   knomura   quick   STDIN   602 2 16 5gb
00:30 R 00:03
hpc0431/7+hpc0431/6+hpc0431/5+hpc0431/4+hpc0431/3+hpc0431/2+hpc0431/1+hpc0431/0+hpc0432/7+hpc0432/6+hpc0432/5+hpc0432/4+hpc0432/3+hpc0432/2+hpc0432/1+hpc0432/0
7456592.hpc-pbs.   knomura   quick   STDIN   -- 2 16 -- 00:30 Q
--   --
```
Interactive PBS Job

• PBS has a special job submission mode that allows a user to access the allocated computing resources interactively. This is called interactive mode or interactive job.

• New login shell starts on one of the computing nodes once an interactive job is accepted.

• While the interactive job is running, you can log on the computing nodes via ssh.

• You can run program as many as you want until the requested time expires. Extremely useful for compile/debug/test your code.
Interactive PBS Job (cont.)

So, what should I do to use the interactive mode?

-> Add `-l` option to `qsub` command and list all PBS options!

```bash
[hpc-login2 ~]$ qsub -l -d . -l nodes=2:ppn=8
qsub: waiting for job 7456677.hpc-pbs.hpcc.usc.edu to start
qsub: job 7456677.hpc-pbs.hpcc.usc.edu ready

Begin PBS Prologue Sat Mar  1 14:21:49 PST 2014
Job ID:    7456677.hpc-pbs.hpcc.usc.edu
Username:  knomura
...
Nodes:    hpc2015 hpc2016
...
End PBS Prologue Sat Mar  1 14:22:00 PST 2014
```

---

So, what should I do to use the interactive mode?

-> Add `-l` option to `qsub` command and list all PBS options!
Interactive PBS Job (cont.)

• While an interactive job running, you can open another terminal, login to fileserver, then login to the allocated nodes for the interactive job.
• Very handy to check if your job is running as you specified in your PBS script.

```
[hpc-login2 ~]$ ssh hpc2015
Last login: Fri Feb 28 13:56:07 2014 from hpc-login2-l.hpcc.usc.edu
[hpc2015 ~]$ cat /proc/cpuinfo
processor : 0
...
model name : Intel(R) Xeon(R) CPU X5650 @ 2.67GHz
...
[hpc2015 ~]$ head /proc/meminfo
MemTotal: 24596976 kB
MemFree: 23240620 kB
...
[hpc2015 ~]$ top
```
Outline

1. HPC Overview
2. Portable Batch System (PBS)
   - PBS Basics
   - Interactive Mode
   - Job Monitoring
3. Account Management
   - Directory and Quota
   - Computing Time
4. Software Repository
Directory and Quota

Each user has three types of directories, home, project and staging directories. There are limits on max. number of files and max. size of total data you can save. They are called file quota and disk quota respectively.

• **Home directory**
  Default directory. When you login to the HPC cluster will be in home directory. Home directory looks like /home/rcf-xx/username. Home directory has 1 GB of disk quota and 100,000 files of file quota.

• **Project directory**
  Each user will have one directory for each project you belong to. Project directory looks like /home/rcf-proj/projectid. Quota on project directory varies depending on each project. Project directory has backup.
Directory and Quota

- Staging directory
  Like project directory, staging directory is created for each project. Path looks like `/staging/projectid/username`.

Pros: No quota on disk nor file. Staging directory deploys the latest parallel file system (OrangeFS).

Cons: NO DATA BACKUP, and all files will be cleaned up every downtime (twice a year).

Good for applications with high-frequency data access (read and write). After your calculation finished, you should move results to your project directory.
/staging Demo

Benchmark Procedure:
Get an Infiniband node. Use `dd` command to measure the speed of a 1GB write on project directory and staging directory.

```
[~]$ qsub -l -d . -l nodes=1:IB
[knomura@hpc3000 ~]$ cd /home/rcf-proj/hpcc/knomura/
[knomura@hpc3000 knomura]$ dd if=/dev/zero of=zero bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 9.8217 s, 109 MB/s

[knomura@hpc3000 knomura]$ cd /staging/hpcc/knomura/
[knomura@hpc3000 knomura]$ dd if=/dev/zero of=zero bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 1.35759 s, 791 MB/s
```
Monitoring Your Quota: myquota

myquota shows the quota on your home and project directories.

[~]$ myquota
--------------------------------------------
Disk Quota for /home/rcf-12/knomura ID 55302

<table>
<thead>
<tr>
<th>Used</th>
<th>Soft</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>36960</td>
<td>100000</td>
</tr>
<tr>
<td>Bytes</td>
<td>2.98G</td>
<td>4.30G</td>
</tr>
</tbody>
</table>

--------------------------------------------
Disk Quota for /home/rcf-proj/hpcc ID 419

<table>
<thead>
<tr>
<th>Used</th>
<th>Soft</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>417850</td>
<td>1000000</td>
</tr>
<tr>
<td>Bytes</td>
<td>264.29G</td>
<td>400.00G</td>
</tr>
</tbody>
</table>

Files for file quota and Bytes for disk quota.

Hard quota is the absolute limit you can store.
Monitoring Your Quota: myquota

• Your job may crash due to the quota, failed to write files in either home directory or project directory.

• Why home directory? If you don’t specify where PBS output file will be stored in your PBS script, it may try to store the output file in your home directory and crash since no quota.

• Pay attention to file quota (max number of files) also. Some users have millions of tiny files!

• If you need more space in project directory, submit a request from your project page.

https://www-rcf.usc.edu/rcfdocs/hpcc/allocations/
Computing Time

• To be able to run your job on the HPC cluster, you need to have computing time (unit is core \( \times \) hour) in your project account.

• Whenever your job finishes (successfully or unsuccessfully), the project account is charged by the number of processors \( \times \) wallclock time your job spent.

• If you request 2 nodes with 4 processors per nodes for 2 hours (\(-l\) nodes=2:ppn=4,walltime=2:00:00), the total charge is \( 2 \times 4 \times 2 = 16 \) core-hour.
Monitoring Computing Time: mybalance

mybalance shows current balance of project account

[~]$ mybalance -h
Balance    Name
---------- -----
20000.00   lc_kn1

- All users have default account and computing time will be charged on the default account automatically.
- Sometimes you need to specify account name in your PBS script by -A option. E.g. -A lc_kn1
- If your job doesn’t start and stay in queue long time, always good idea to check if your project has enough balance.
Outline

1. HPC Overview
2. Portable Batch System (PBS)
   - PBS Basics
   - Interactive Mode
   - Job Monitoring
3. Account Management
   - Directory and Quota
   - Computing Time
4. Software Repository
HPC install & maintain software in a single software repository.

**Compilers:** gnu, intel, pgi

**Numerical Libraries:** mpich, openmpi, cuda, fftw, petsc

**Molecular Simulation:** NAMD, gromacs, amber

**Quantum Chemistry:** gaussian, schroderinger

**Numerical Environment:** matlab, R, python

hpc-login1.usc.edu for 32-bit applications
hpc-login2.usc.edu for 64-bit applications
Software Repository: /usr/usc

How does the software repo look like?

[~]$ cd /usr/usc/
[~]$ ls -F
acml/  dict@  gurobi/ leveldb/ NAMD/ qesperso/
amber/  etc/  hadoop/ libroadrunner/ netcdf/ qiime/
bbcp/  fdtd/  hdf5/  llvm/  nwdb/  R/  bin/  fftw/
hdfview@ magma/  opencv/  root/  boost/  gaussian/  hpctoolkit/
mathematica/  openmpi/  sas/  caffe/  gflags/  imp/  matlab/
....

[~]$ ls -F matlab/
2009a/ 2009b/ 2010b/ 2011a/ 2013a/ 2013b/ 2014a/ default@ Scripts/
Software Repository: /usr/usc

How can I use software?
- First, go to the directory of software you want to use. Usually each software has several subdirectories for different versions. Pick one you want to use.
- Look for setup scripts. setup.sh for bash users and setup.csh for tcsh users.
- Source the setup file!
  
  [~]$ source /usr/usc/matlab/2014a/setup.sh
  [~]$ source /usr/usc/matlab/2014a/setup.csh
What will happen by sourcing a setup script?

```
[~]$ cat /usr/usc/matlab/2014a/setup.sh
if [ ! "${USCENV_MATLAB}" ]; then
  USCENV_MATLAB="1"
  PATH=/usr/usc/matlab/2014a/bin:$PATH
  export USCENV_MATLAB PATH
# License file config is in licenses/network.lic
if [ ! "$MATLAB_ARCH" ];then
  export MATLAB_ARCH=glNxa64
fi
# Remote OpenGL calls over X require this
LIBGL_ALWAYS_INDIRECT=1
export LIBGL_ALWAYS_INDIRECT
fi
[~]$ echo $PATH
/usr/usc/matlab/2014a/bin:/usr/local/bin:/bin:/usr/bin:/sbin
```
A Case Study: System vs Software Repo

- Sometimes software and libraries (e.g. gcc, python, fftw) **come with OS**
- Although command name is the same, the system software and repo software are often different (versions, libraries, developers). Make sure that you use what you want to use
- **which** command shows the absolute path of a command

```
[~]$ which python
/usr/bin/python
[~]$ source /usr/usc/python/enthought/default/setup.sh
[~]$ which python
/usr/usc/python/enthought/default/bin/python
```
Software Installation

• There are many open source software and libraries available for Linux

• Open source applications are often provided as a tarball that includes source codes, scripts and instructions to compile and install it

  .configure && make && make install

• Proprietary software can be installed on the HPCC cluster (e.g. Intel&PGI compilers, Matlab, Mathematica, lumerical FDTD) but doing so requires a proper license

• Windows applications are not supported
Example: python Installation

Installation Steps
1. Download source code from python website
2. Compile & install under project directory
3. Give proper permissions to share with group member
4. Set environment variables
Example: python Installation

```
[Python-2.7.6]$ less README
[Python-2.7.6]$ ./configure --help
`configure' configures python 2.7 to adapt to many kinds of systems.
Usage: ./configure [OPTION]... [VAR=VALUE]...
Configuration:
  -h, --help       display this help and exit
    --help=short   display options specific to this package
    --help=recursive display the short help of all the included packages
  -V, --version    display version information and exit
  -q, --quiet, --silent do not print `checking ...' messages
    --cache-file=FILE cache test results in FILE [disabled]
  -C, --config-cache alias for `--cache-file=config.cache'
```

After expanding the tarball, it’s always a good idea to take a look at README and help messages from the configure script before deploying.
# Change to project directory
cd /home/rcf-proj/projectid/myacctn
# Download python tarball
wget http://www.python.org/ftp/python/2.7.6/Python-2.7.6.tgz
# Unarchive the tarball and change directory
tar xvfz Python-2.7.6.tgz & & cd Python-2.7.6
# run configure script with custom installation directory
./configure --prefix=/home/rcf-proj/projectid/myacctn/python/2.7.6/
# Compile and install
make & & make install
# Create setup script
vi setup.sh (or setup.csh)
# Check the setup script
source setup.sh
echo $PATH
# Set the readable & executable permission for your group
chmod -R g+rx .

prefix=/home/rcf-proj/projectid/myacctn/python/2.7.6/
export PATH=${prefix}/bin:$PATH (bash)
setenv PATH=${prefix}/bin:$PATH (tcsh)
Want to learn more?

- Up and Running with Bash Scripting
- Unix for Mac OS X Users
- Using Regular Expressions
- Perl 5 Essential Training
- R Statistics Essential Training
- C/C++ Essential Training
- Up and Running with Python
- Python 3 Essential Training
- Up and Running with MATLAB
- Up and Running with R and More!

From: Unix for Mac OS X Users
Want to learn more?

http://software-carpentry.org

<table>
<thead>
<tr>
<th>Version 5</th>
<th>Version 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>This material is what we currently use in bootcamps. View source on GitHub</td>
<td></td>
</tr>
<tr>
<td>This material was created in 2010-11, and is now in maintenance mode. View source on GitHub</td>
<td></td>
</tr>
</tbody>
</table>
| • Introduction  
• The Unix Shell  
• Version Control with Git  
• Programming with Python  
• Using Databases and SQL  
• A Few Extras  
• Instructor's Guide  
• Setup Instructions  
• Recommended Reading  
• Glossary  
• Our Team |
| • Using Subversion  
• The Unix Shell  
• Programming in Python  
• Testing  
• Sets and Dictionaries  
• Regular Expressions  
• Databases  
• Using Access  
• Data Management  
• Object-Oriented Programming  
• Program Design  
• Make  
• Systems Programming  
• Spreadsheets  
• Matrix Programming with NumPy  
• MATLAB  
• Multimedia Programming  
• Software Engineering  
• Essays |

Who We Are

Our volunteers teach basic software skills to researchers in science, engineering, and medicine. Founded in 1999, we are the Mozilla Science Lab.

Find a bootcamp
Thank you!