Introduction to HPC Cluster Computing

Avalon Johnson

Center for High-Performance Computing

Outline

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

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)

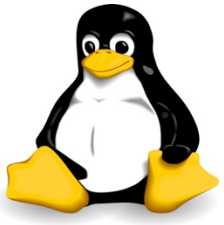
Software Service

- 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.

amber	fftw	cuda	intel	gnu	sas
python	Pegasus	matlab	fdtd	mathematica	iperf
qespresso	hdf5	globus	gaussian	pgi	spss
qiime	libroadrunner	hadoop	gromacs	llvm	stata
R	mpich	hdfview	lammps	boost	taxila
git	openmpi	hpctoolkit	NAMD	cellprofiler	bbcp
opencv	papi	petsc	schrodinger	gurobi	caffe

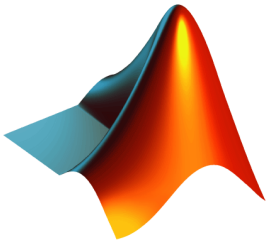
HPC trainings and workshops

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



OpenACC
DIRECTIVES FOR ACCELERATORS

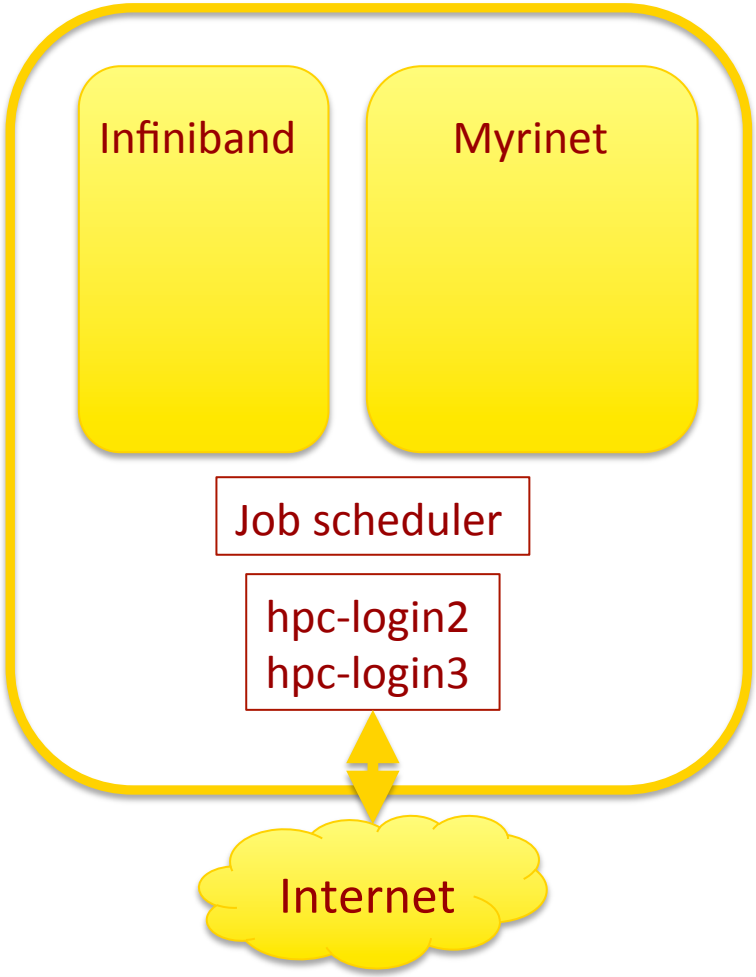
NVIDIA



GPU programming workshop by NVIDIA

HPC node organization

- Compute nodes are connected by two high speed low latency networks, Infiniband and Myrinet
- The Infiniband and the Myrinet networks are NOT connected to each other
- Nodes connected via the Infiniband network CANNOT do HIGH SPEED communication with nodes in the Myrinet network
- hpc-login2 and hpc-login3 provide access from the outside to the nodes.



Outline

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

Directories and Quotas

Each user has two types of permanent directories: home and project.

Each user also has access to a 'quasi permanent' staging directory. While running a job, a user also has access to a 'temporary' scratch directory.

- **Home directory**

This is your default directory. When you login to the HPC cluster you will be in your home directory.

$ cd

$ pwd

/home/rcf-40/avalonjo

Directories and Quotas

- **Project directory**

Each user has one directory for each project that you belong to. Each project directory is of the form:

/home/rcf-proj/<projectid>/<userid>

Group for CSC596 is lc_an2

$ groups
g03 lc_an2 hpcusers

$ ls -ld /home/rcf-proj/an2/avalonjo

drwx------ 28 avalonjo lc_an2 4096 Mar 30 14:41 /home/rcf-proj/an2/avalonjo

Directories and Quotas

- **Staging directory**

Like the project directory, each user has one staging directory for each project that they are in.
Each staging directory is of the form:

/staging/<projectid>/<userid>

$ groups
g03 src ucsadmin rds lc_test gaussian hpcusers lc_hpcc

$ ls -ld /staging/an2/avalonjo

drwxr-s--- 1 avalonjo lc_an2 4096 Feb 26 17:17 /staging/an2/avalonjo/

Directories and Quotas

The project and home directories both have limits on usage called quotas. These quotas apply BOTH to the number of files as well as total disk space used.

- **Home directory**

  Home directory has 1 GB of disk space quota and 100,000 files of file quota.

- **Project directory**

Your project usage limits is dictated by the project itself.
lc_an2 has a limit of 500GBytes

Directories and Quotas

- **Staging directory**

The staging directories have No quotas on disk space or number of files. Is a parallel file system (OrangeFS)

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

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

Directories and Quotas

- **/scratch directory**

The /scratch directory is created for each job and is comprised of all the 'free' disk space present in the first 20 nodes in a job. It is created using a parallel file system (OrangeFS).

This space is available to ALL the nodes running your job.

This size will vary depending on the size of the jobs and nodes but will normally vary between about 1TB (for a one node job) and 20TB (for jobs > 20 nodes.)

NO DATA BACKUP, and all files will be cleaned up at job completion.

Monitoring Your Quota: myquota

myquota shows the quota on your home and project directories.

```
$ myquota
----------------------------------------
Disk Quota for /home/rcf-40/avalonjo ID 203387

             Used       Soft       Hard
    Files    9501       100000     101000
    Bytes    721.41M    1.00G      1.00G
----------------------------------------
Disk Quota for /home/rcf-proj2/hpcc ID 419

             Used       Soft       Hard
    Files    502016     1000000    1100000
    Bytes    433.86G    500.00G    502.00G
----------------------------------------
```

Files for file quota and **Bytes** for disk quota.

Hard quota is the absolute limit you can store.

Monitoring Your Quota: myquota

- If you go over quota your job may crash when it fails to write files. This can be in either home directory or project 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 if you are over quota.
- Pay attention to files quota (number of files). Some users have millions of tiny files. This places a very large burden on the system since these all have to be backed up!
- 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 #cores × hr) in your project account.
- Whenever your job finishes (successfully or unsuccessfully), the project account is charged by the number of cores × 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 2x4x2 = 16 core-hours.

Monitoring Computing Time: mybalance

mybalance shows current balance of project account

```
$ mybalance
Balance  Name
-------- ----------------
Infinity hpccadm
  227032 HPCCTestFund
Infinity HPCWorkShopApr2015
```

- 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, remaining in the queue a long time, it's always a good idea to check if your project has enough balance.

Outline

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

Software Repository: /usr/usc

HPC installs & maintains 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, schrodinger*
Numerical Environment: *matlab, R, python*

hpc-login2.usc.edu for 64-bit applications
hpc-login3.usc.edu for 64-bit applications

Software Repository: /usr/usc

What does the software repository look like?

```
$ cd /usr/usc/
$ ls –F
acml/       fftw/       imp/        mpich2/     qespresso/
amber/      gaussian/   intel/      mpich-mx/   qiime/
aspera/     gflags/     iperf/      mvapich2/   R/
bbcp/       git/        java@       NAMD/       root/
bin/        globus/     jdk/        ncview/     sas/
….
$ ls -F hello_usc
1.0/  2.0/  3.0/
```

Software Repository: /usr/usc

How can I access software?

- First, go to the directory of the software you want to use. Usually each software has several subdirectories for different versions. Pick the one you want.
- Look for setup scripts: setup.sh for bash users and setup.csh for tcsh users.
- source the setup file!

  $ source /usr/usc/hello_usc/2.0/setup.sh

  $ source /usr/usc/hello_usc/3.0/setup.csh

Software Repository: /usr/usc

What will happen when I source a setup script?

```
$  hello_usc
-bash: hello_usc: command not found

$ source /usr/usc/hello_usc/2.0/setup.sh

$ hello_usc

   Hello USC!!!.
   I am version 2.0 running on host: hpc-login3


$ which hello_usc
/usr/usc/hello_usc/2.0/bin/hello_usc
```

Software Repository: /usr/usc

```
$ cat /usr/usc/hello_usc/2.0/setup.sh
 if [ "x" = "x$USCENV_HELLO_USC" ];then
        USCENV_HELLO_USC=1
        HELLO_PREFIX=/usr/usc/hello_usc/2.0
        export USCENV_HELLO_USC

        if [ "x${PATH}" = "x" ]; then
                PATH="${HELLO_PREFIX}/bin:/bin:/usr/bin:/usr/local/bin"
        else
                PATH=${HELLO_PREFIX}/bin:$PATH
        fi
fi
```

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

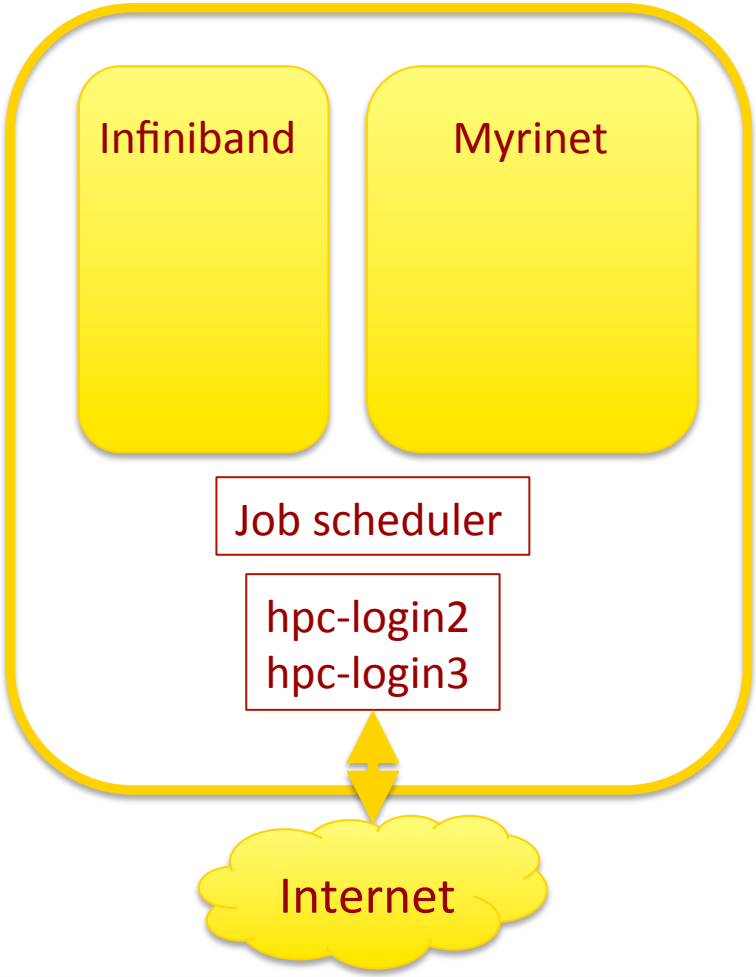
```
$ which python
/usr/bin/python
$ source /usr/usc/python/enthought/default/setup.sh
$ which python
/usr/usc/python/enthought/default/bin/python
```

Outline

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

Portable Batch System (PBS)

- You want to have compute nodes assigned to you
- To get compute nodes assigned to you you will need to run the qsub command
- The qsub command is your interface into the Job scheduler, which finds unused nodes and assigns them to you based on your requirements
- If there are no free nodes your JOB get's queued waiting it's turn



PBS Commands: qsub

qsub	submit a job to computing cluster
-l	resource list
nodes	number of nodes
ppn	processor per core (nodes attribute)
gpus	GPU node request (nodes attribute)
nodeattr	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

Interactive PBS Jobs

- PBS has a special job submission mode that allows a user to access 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 your assigned computing nodes via ssh.
- You can run programs as many times as you want until the requested time expires. Extremely useful for compile/debug/test your code.

Interactive PBS Jobs (cont.)

-> Add –I (eye) option to qsub command

```
hpc-login3: qsub –d  -I 'nodes=2:ppn=8' -I
qsub:   waiting for job 11785338.hpc-pbs.hpcc.usc.edu to start
qsub:   job 11785338.hpc-pbs.hpcc.usc.edu ready
----------------------------------------
Begin PBS Prologue Wed Apr  1 17:07:15 PDT 2015
Job ID:                 11785338.hpc-pbs.hpcc.usc.edu
Username:               avalonjo
…
Nodes:                  hpc2062 hpc2597
PVFS:                   /scratch (98G), /staging (328T)
TMPDIR:                 /tmp/11785338.hpc-pbs.hpcc.usc.edu
…
----------------------------------------
hpc2597: ssh hpc2062
hpc2062: hostname
hpc2062
```

Interactive PBS Job (cont.)

- While an interactive job is running, you can open another terminal, log in to headnode, then log in 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-login3: ssh hpc2062
Last login: Wed Apr  1 17:07:52 2015 from hpc2597-e0.hpcc.usc.edu

hpc2062: head -10 /proc/cpuinfo
processor: 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 15
model name      : Intel(R) Xeon(R) CPU          E5345  @ 2.33GHz

hpc2062: head -2 /proc/meminfo
MemTotal:      12191088 kB
MemFree:       10876384 kB
```

Interactive PBS Job (cont.)

- You can now test your commands to see how they will run.

```
hpc-login3: qsub ' –l 'nodes=2:ppn=4’ -l ‘walltime=2:00:00’ –A lc_an2 -I
qsub: waiting for job 11788009.hpc-pbs.hpcc.usc.edu to start
…
End PBS Prologue Thu Apr  2 10:06:10 PDT 2015
----------------------------------------
hpc2062: source /usr/usc/hello_usc/2.0/setup.sh
hpc2062: which hello_usc
/usr/usc/hello_usc/2.0/bin/hello_usc

hpc2062: pbsdsh -u /usr/usc/hello_usc/2.0/bin/hello_usc

   Hello USC!!!.
   I am version 2.0 running on host: hpc2062

   Hello USC!!!.
   I am version 2.0 running on host: hpc2081

# Try without the –u, what happens?
```

Interactive PBS Job (cont.)

- Let's compile and run an OpenMPI program.

```
hpc2062: cd /home/rcf-proj/hpcc/avalonjo
hpc2062: mkdir Tmp
hpc2062: cd Tmp

hpc2062: cp /home/rcf-proj/hpcc/WorkshopFiles/helloWorldMPI.c .
hpc2062: cp /home/rcf-proj/hpcc/WorkshopFiles/compile.sh .
hpc2062: cat compile.sh
#!/bin/sh

CC=mpicc make helloWorldMPI

hpc2062: source /usr/usc/openmpi/1.8.4/setup.sh

hpc2062: ./compile.sh
mpicc     helloWorldMPI.c   -o helloWorldMPI

hpc2062: ls
compile.sh*  helloWorldMPI*  helloWorldMPI.c
```

Interactive PBS Job (cont.)

```
hpc2062: which mpiexec
/usr/usc/openmpi/1.8.4/bin/mpiexec

hpc2062: mpiexec ./helloWorldMPI
Hello World from rank 1 running on hpc2062!
Hello World from rank 2 running on hpc2062!
Hello World from rank 3 running on hpc2062!
Hello World from rank 0 running on hpc2062!
MPI World size = 8 processes
Hello World from rank 4 running on hpc2081!
Hello World from rank 5 running on hpc2081!
Hello World from rank 6 running on hpc2081!
Hello World from rank 7 running on hpc2081!
```

filesystem benchmarks (demo)

Benchmark Procedure:

Use dd command to measure the speed of a 1GB write on project,scratch and staging directory.

$ qsub -l 'nodes=2:ppn=16:IB' -l 'walltime=2:00:00' –A lc_an2 -I

…

Nodes: hpc3260 hpc3261

…

hpc3260: df

Filesystem 1K-blocks Used Available Use% Mounted on

…

tcp://hpc-ofs03.ib.hpcc.usc.edu:3334/staging
351541493760 129451282432 222090211328 37% /staging

tcp://hpc3260.ib.hpcc.usc.edu:3334/pvfs2-fs
1781469184 1458176 1780011008 1% /scratch

almaak-08:/export/samfs-proj2/proj
171885621248 60336051072 111549570176 36% /auto/rcf-proj

filesystem benchmarks (demo)

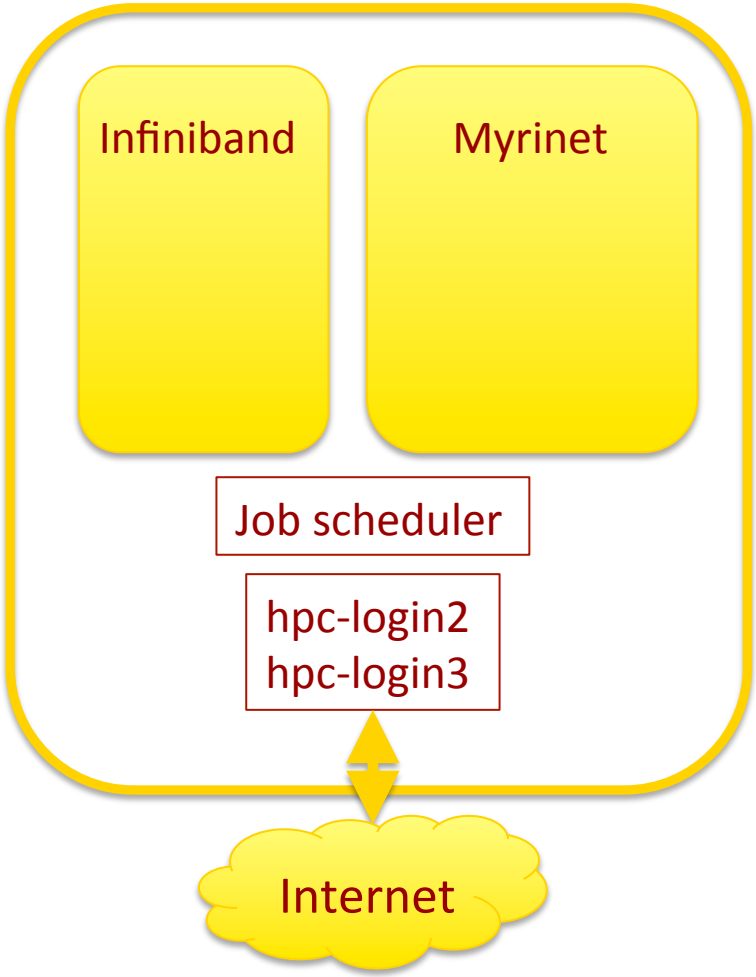
hpc3260: cd /staging/hpcc/avalonjo/tmp
hpc3260: dd if=/dev/zero of=fileOfzeros bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 2.69193 s, 399 MB/s

hpc3260: cd /scratch
hpc3260: dd if=/dev/zero of=fileOfzeros bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 2.48319 s, 432 MB/s

hpc3260: cd /home/rcf-proj/hpcc/avalonjo/tmp
hpc3260: dd if=/dev/zero of=fileOfzeros bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 9.82488 s, 109 MB/s

Portable Batch System (PBS)

- To submit your job to the cluster, create a text file which describes the 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, then the job scheduler will start your job.



PBS script: Example

On hpc-login3 using the nano editor create helloworld.PBS in your Tmp directory. Then submit using qsub helloworld.PBS

```
1  #!/bin/bash
2  #PBS -l nodes=1:ppn=8
3  #PBS -l walltime=00:10:00
4  # Next 2 lines for HPC workshop only
5  #PBS –A lc_an2
6  #PBS –N CSC596Example
7  # change to your project directory
8  cd /home/rcf-proj/hpcc/avalonjo/Tmp
9
10 # source setup file (setup.csh for tcsh)
11 source /usr/usc/openmpi/1.8.4/setup.sh
12
13 # run command
14 mpiexec helloWorldMPI
```

1: Set up which shell to use
2: one node with 8 procs per node
3: request for 10 minutes
5: account lc_an2
6: name of session
8: cd to project dir.
9: blank
10: comment
11: Source setup file to use openmpi (gnu version)
12: blank
13: comment
14: run helloWorldMPI

PBS script: a bit more advanced

```
#!/bin/bash
#PBS -l nodes=4:ppn=16:gpus=2,pvmem=2GB  # job needs 2G per ppn
#PBS -l walltime=24:00:00
#PBS –m abe                              # email sent on abort/begin/end
#PBS –M avalonjo@usc.edu                 # my email address
#PBS -A lc_an2
#PBS –d /home/rcf-proj/hpcc/avalonjo    # change into this directory
#PBS -N my_mpicode                       #Name of my job

# 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
```

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 jobs.
- The job scheduler automatically selects which queue to be assigned on your job depending on the your request. No need to specify queue by users.

Queue Name	Maximum Jobs Queued	Maximum Node Count	Maximum Wall Time	Maximum Jobs per User
main	1000	99	24 hours	10
quick	100	4	1 hour	10
large	100	256	24 hours	1
largemem	100	1	336 hours	1

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

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

```
#PBS -N myjob
#PBS -d /home/rcf-proj/hpcc/avalonjo
#PBS -l pmem=1gb
#PBS –A workshop
```

```
#PBS -l nodes=16:ppn=12
#PBS -l walltime=23:00:00
#PBS -A workshop
#PBS -d .
```

node attribute: nodetype

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

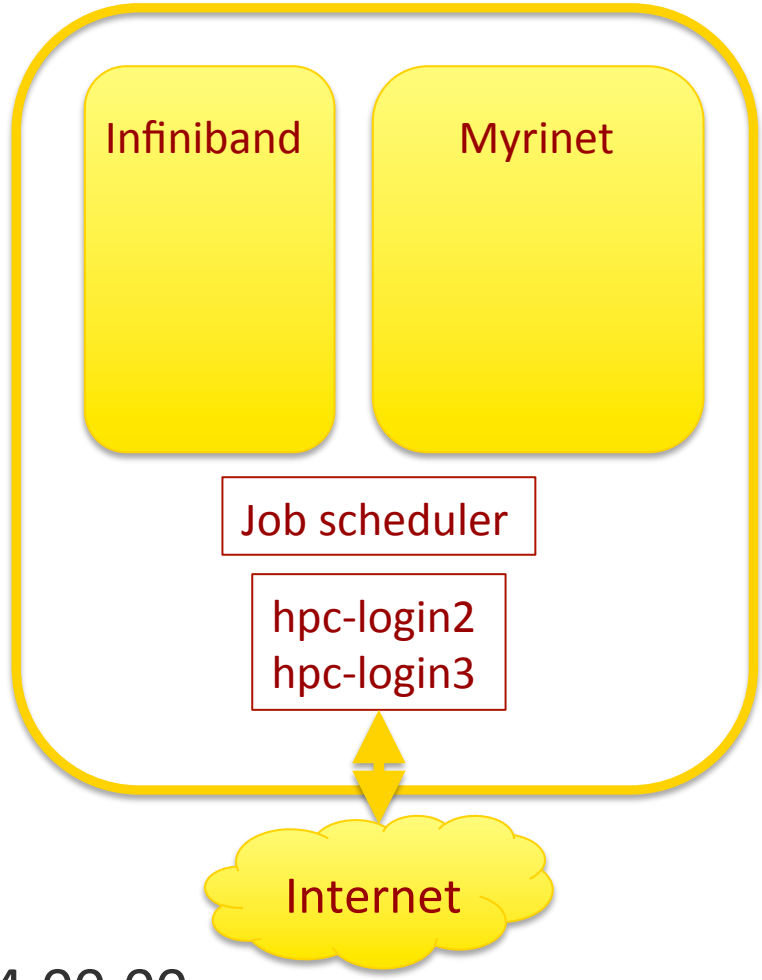
$ qsub -I -d . -l nodes=2:ppn=8:pe1950

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

http://hpcc.usc.edu/support/infrastructure/node-allocation/

node attribute: myri and IB

- As previously mentioned, there are two different interconnects in the HPC cluster, called Myrinet and Infiniband.
- The Myrinet and Infiniband networks are not connected to each other.
- If not specified the system will use the set of nodes that allow your job to start.
- Codes compiled to use MPICH will only run on the Myrinet nodes.
- OpenMPI codes will run on either.



qsub -l nodes=10:ppn=8:myri,walltime=4:00:00
qsub -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

```
$ qstat -u avalonjo
hpc-pbs.hpcc.usc.edu:
                                                                  Req'd   Req'd      Elap
Job ID                  Username  Queue  Jobname  SessID NDS  TSK  Memory  Time   S  Time
----------------------- --------- -------- -------------- ------ ----- ------ ------ --------- - ---------
9667416.hpc-pbs.  avalonjo main   testjob    3827  10   20   --    6:00:00  R  3:21:28
```

Job Monitoring: qstat

```
$ qstat main | head
Job ID                  Name             User            Time Use S Queue
----------------------- ---------------- --------------- -------- - -----
9043999.hpc-pbs         job1             user1                 0 Q main
9447030.hpc-pbs         Job2             user2                 0 Q main
9629959.hpc-pbs         ...job.pbs       user3                 0 Q main
9629975.hpc-pbs         ...job.pbs       user3                 0 Q main
9633223.hpc-pbs         job3             user4                 0 Q main
9653476.hpc-pbs         ...job.pbs       user3                 0 Q main
9676843.hpc-pbs         test.pbs         user5          169:54:2 R main
9679200.hpc-pbs         rsync            user6         10:17:15 R main
```

Job Monitoring: myqueue

myqueue — Display jobs status and allocated node list for your running jobs.

```
hpc-login3: myqueue

hpc-pbs.hpcc.usc.edu:
                                                                  Req'd       Req'd       Elap
Job ID               Username    Queue    Jobname       SessID  NDS   TSK   Memory   Time    S  Time
-------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
11788258.hpc-pbs.hpcc.  avalonjo    main    STDIN          32291    2     8    --   02:00:00 R  00:30:44
hpc2062/0+hpc2062/1+hpc2062/2+hpc2062/3+hpc2081/0+hpc2081/1+hpc2081/2+hpc2081/3

hpc-login3:
```

Job Monitoring: showstart

showstart Displays approximate start time for your job.
checkjob Displays certain system properties for your job

```
# Look for 'HOLDs' on your job

hpc-login3: checkjob 11788258
job 11788258
…
WallTime:   00:33:28 of 2:00:00
SubmitTime: Thu Apr  2 10:37:24
…
StartPriority:  1
Reservation '11788258' (-00:33:28 -> 1:26:32  Duration: 2:00:00)
```

```
$ showstart   11788258
# Place this in your .bashrc file export CLIENTTIMEOUT='00:10:00'
```

Some Basic Linux Commands

ls	List file and/or directory names
pwd	Print working (current) directory

```
$ touch emptyfile
$ ls -l
total 0
-rw-rw-r--. 1 avalonjo avalonjo 0 Mar 24 13:44 emptyfile
$ touch emptyfile2
$ ls -lt
total 0
-rw-rw-r--. 1 avalonjo avalonjo 0 Mar 24 13:44 emptyfile2
-rw-rw-r--. 1 avalonjo avalonjo 0 Mar 24 13:44 emptyfile
$ pwd
/home/avalonjo/workshop
```

Basic Commands (cont)

mkdir/rmdir	Create/remove directory
cd	Change directory

```
$ ls
emptyfile1  emptyfile2
$ mkdir subdirectory1
$ ls
emptyfile1  emptyfile2  subdirectory1/
$ cd subdirectory1/
$ pwd
/home/avalonjo/workshop/subdirectory1
$ cd ..
$ rmdir subdirectory1
$ ls
emptyfile1  emptyfile2
```

Basic Commands (cont)

cp/mv/rm Copy/move/remove file or directory

alias Create an alias for a command

```
$ ls
emptyfile1  emptyfile2

$ cp emptyfile2 emptyfile3
$ ls
emptyfile1  emptyfile2  emptyfile3

$ mv emptyfile3 emptyfile3_withNewName
$ ls
emptyfile1  emptyfile2  emptyfile3_withNewName
```

Basic Commands (cont)

cp/mv/rm	Copy/move/remove file or directory
alias	Create an alias for a command

```
$ ls
emptyfile1  emptyfile2   emptyfile3_withNewName
$ rm emptyfile1
$ ls
emptyfile2  emptyfile3_withNewName

$ alias rm="/bin/rm -i”    #csh  alias rm ‘/bin/rm –I’
$ rm emptyfile2
/bin/rm: remove regular empty file `emptyfile2'? n
$ mkdir subdir1
```

Basic Commands (cont)

mkdir — Create/remove directory with options

```
$ mkdir subDirectoryLevel1
$ ls
emptyfile2 emptyfile3_withNewName subdir1/ subDirectoryLevel1/
$ ls subDirectoryLevel1
$ mkdir -p subDirectoryLevel1/subDirectoryLevel2/{1,2,3}
$ ls subDirectoryLevel1
subDirectoryLevel2/
$ ls subDirectoryLevel1/subDirectoryLevel2/
1/  2/  3/
```

Basic Commands (cont)

Output redirection Redirects output from command

```
$ ls
emptyfile2  emptyfile3_withNewName  subdir1/  subDirectoryLevel1/
$ ls > output_of_ls
$
```

Basic Commands

cat/more/less Display file contents

```
$ cat output_of_ls
emptyfile2 emptyfile3_withNewName output_of_ls subdir1/ subDirectoryLevel1/
$
```

NAME

less - opposite of more

…

DESCRIPTION

Less is a program similar to more (1), but which allows backward movement in the file as well as forward movement. Also, less does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like vi (1).

Basic Commands (cont)

man — Online manual

man(1) man(1)

NAME
man - format and display the on-line manual pages

SYNOPSIS
man [**-acdfFhkKtwW**] [**--path**] [**-m** system] [**-p** string] [**-C** config_file] [**-M** pathlist] [**-P** pager] [**-B** browser] [**-H** htmlpager] [**-S** section_list] [section] name ...

DESCRIPTION
man formats and displays the on-line manual pages. If you specify section, **man** only looks in that section of the manual. name is normally the name of the manual page, which is typically the name of a command, function, or file. However, if name contains a slash (/) then **man** interprets it as a file specification, so that you can do **man ./foo.5** or even **man /cd/foo/bar.1.gz.**

See below for a description of where **man** looks for the manual page files.

MANUAL SECTIONS
The standard sections of the manual include:

1 User Commands

2 System Calls

3 C Library Functions

4 Devices and Special Files

5 File Formats and Conventions

6 Games et. Al.

7 Miscellanea

8 System Administration tools and Deamons

Distributions customize the manual section to their specifics, which often include additional sections.

OPTIONS
-C config_file
Specify the configuration file to use; the default is **/etc/man.config**. (See **man.config**(5).)

-M path
Specify the list of directories to search for man pages. Separate the directories with colons. An empty list is the same as not specifying **-M** at all. See **SEARCH PATH FOR MANUAL PAGES.**

-P pager
Specify which pager to use. This option overrides the **MANPAGER** environment variable, which in turn overrides the **PAGER** variable. By default, **man** uses **/usr/bin/less -is.**

-B Specify which browser to use on HTML files. This option overrides the **BROWSER** environment variable. By default, **man** uses **/usr/bin/less**-is,

-H Specify a command that renders HTML files as text. This option overrides the **HTMLPAGER** environment variable. By default, **man** uses **/bin/cat,**

-S section_list
List is a colon separated list of manual sections to search. This option overrides the **MANSECT** environment variable.

Bash Config Files

- Configuration files are used to set up user environments, for example, command prompts, path, alias, and so on. Sometimes these are called “dot files”
- .bash_profile and .bashrc are stored in each user’s home directory
- When bash is invoked as a login shell, it first reads /etc/profile. if that file exists, then looks for .bash_profile and .profile

Csh & Tcsh Config Files

- .login & .cshrc are in each user's home directory
- When tcsh is invoked as a login shell, it reads first .tcshrc or, if .tcshrc is not found, .cshrc, then .history, then .login, and finally .cshdirs
- When tcsh is invoked as a non-login shell, it only reads /etc/csh.cshrc and .cshrc

Permission & Ownership

- File and directory have ownership and permission
- Three types of permission, readable, writeable and executable
- Each permission is given to three groups, owner, group and others

```
$ ls -l output_of_ls
-rw-rw-r--. 1 avalonjo avalonjo 95 Mar 24 14:12 output_of_ls
```

r readable, **w** writable, **x** executable

u user (owner), **g** group, **o** others, **a** all

Permission & Ownership

chmod	Change file/directory permission
chgrp grp file	Change group that file belongs to
chmod a+w file	Add W permission to all users
chmod o-rwx file	Remove R/W/E permission from others
chmod 750 file	Add R/W/E gives permission to user, R/E gives permission to group but no permission to others

7 = rwx, 5 = r-x, 0 = --- therefore 750 = rwxr-x---

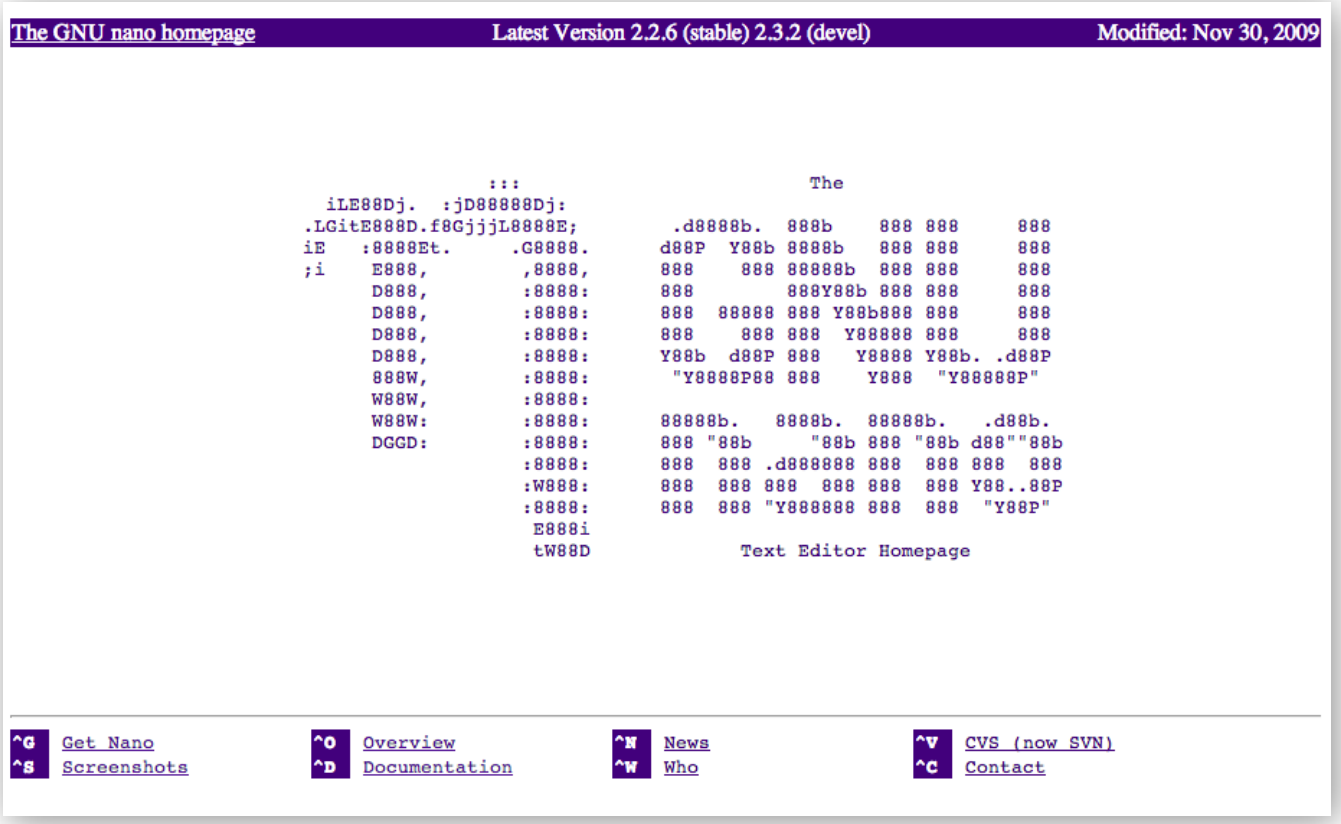
r(4) readable, **w(2)** writable, **x(1)** executable

u user (owner), **g** group, **o** others, **a** all

Summary

ls	List file and/or directory names
pwd	Print working (current) directory
mkdir/rmdir	Create/remove directory
cd	Change directory
cp/mv/rm	Copy/move/remove file or directory
cat/more/less	Display file contents
man	Display online manual
chmod/chown	Change permission/ownership

Text Editor: GNU nano



http://www.nano-editor.org/

nano basics (cont.)

Arrow-keys	Move cursor
Enter	Change line
CTRL+a	Move to the beginning of line
CTRL+e	Move to the end of line
CTRL+v	Move forward one page
CTRL+y	Move backward one page

nano basics (cont.)

CTRL+o	Save file
CTRL+w	Search text
CTRL+d	Delete a character
CTRL+k	Remove a line
CTRL+u	Paste buffer
CTRL+x	Save data and exit

Shell Scripting: hello.sh

```
#!/bin/bash
echo “hello world”
```

1. Open Terminal and type
> nano hello.sh

2. Type text in left box

3. Save it and close nano

4. Add executable permission
> chmod og+x hello.sh

5. Run it
> ./hello.sh

Shell Scripting: clock.sh

```
#!/bin/bash
for n in {0..9};  do
   date +"%r"
   sleep 1
done
```

1. Add executable permission to clock.sh
2. Type ./clock.sh

```
[~]$ ./clock.sh
   02:21:02 PM
   02:21:03 PM
   02:21:04 PM
      …
```

Environmental Variable

Shell (bash or tcsh) adds “environmental” variables to various data, such as host, user, software settings etc.

env display all environmental variables

```
$ env
NNTPSERVER=newshub.ccs.yorku.ca
MANPATH=/usr/local/man:/usr/man:/usr/share/man:/usr/local/share/man:/usr/X11R6/man
HOSTNAME=indigo.usc.edu
TERM=xterm
SHELL=/bin/bash
HISTSIZE=6000
SSH_CLIENT=71.160.93.57 51323 22
USER=avalonjo
MAIL=/var/spool/mail/avalonjo
PATH=/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin
PWD=/home/avalonjo/workshop
EDITOR=vi
LANG=en_US.UTF-8
HOME=/home/avalonjo
LOGNAME=avalonjo
…
```

Environmental Variable (cont.)

One important environmental variable is PATH, which is a list of directories that store commands. Whenever you type a command, your shell looks for the command from directories listed in PATH. If a command is not found in PATH, you have to type absolute path of the commands.

echo display an environmental variable
export (or setenv) set an environmental variable

```
$ echo $PATH
/usr/bin:/usr/bin:/usr/local/bin:/bin:/usr/bin:/sbin
$ export PATH=${PATH}:/home/avalonjo/workshop (bash)
$ setenv PATH ${PATH}:/home/avalonjo/workshop (tcsh)
```

Redirect and Pipe

A special character > redirects output from commands into different channels. Two types of outputs are commonly used, standard output and standard error.

```
[~]$ env                                …
MANPATH=/usr/share/man:
HOSTNAME=hpc-login2
TERM=xterm-256color
SHELL=/bin/bash
HISTSIZE=1000
…
[~]$ env > env.log

[~]$ cat env.log
MANPATH=/usr/share/man:
HOSTNAME=hpc-login2
TERM=xterm-256color
SHELL=/bin/bash
```

Redirect and Pipe (cont.)

A special character | pass output from one command to another command, called pipe. Many command can be daisy-chained by pipe.

grep	print lines matching a pattern
head/tail	show first/last several lines
sort	sort text alphabetically/numerically

Example: Print top 5 users who are consuming CPU except myself

```
[~]$ ps axuw | grep -v ${USER} | sort -r -n -k 3 | head -n 5
```

Process Management

Process is a unit of program. Whenever you run a command, at least one process will be created. Each process is assigned a unique integer called process ID.

top display currently running jobs

```
Processes: 200 total, 2 running, 6 stuck, 192 sleeping, 922 threads                                                 11:29:56
Load Avg: 1.34, 1.21, 1.03  CPU usage: 1.93% user, 4.52% sys, 93.53% idle    SharedLibs: 11M resident, 10M data, 0B linkedit.
MemRegions: 93261 total, 2125M resident, 72M private, 657M shared. PhysMem: 5745M used (1364M wired), 595M unused.
VM: 489G vsize, 1066M framework vsize, 174811(0) swapins, 355954(0) swapouts.
Networks: packets: 38059781/27G in, 20628194/7089M out. Disks: 10560617/208G read, 14170246/446G written.

PID    COMMAND      %CPU TIME     #TH  #WQ  #PORT #MREGS MEM    RPRVT  PURG   CMPRS  VPRVT  VSIZE  PGRP  PPID  STATE    UID
97999  syspolicyd   0.0  00:00.06 2    1    23    40     2824K  2640K  0B     920K   53M    2412M  97999 1     sleeping 0
97926- Google Chrom 0.0  00:23.02 4    0    58    132    8356K  6752K  0B     6304K  106M   824M   97923 97923 sleeping 501
97923- Google Chrom 0.0  00:57.03 34   1    375+  442+   33M+   35M+   0B     11M    322M+  1098M+ 97923 177   sleeping 501
97477  AppleIDAuthA 0.0  00:00.01 3    2    38    40     404K   240K   0B     396K   46M    2412M  97477 177   sleeping 501
97238- Microsoft Po 0.3  11:10.39 11   2    255+  4211+  178M+  121M+  56M    41M    215M+  1527M+ 97238 177   sleeping 501
92796- dbfseventsd  0.0  00:02.79 1    0    7     27     32K    12K    0B     80K    20K    591M   248   89632 sleeping 501
92788- Dropbox      0.1  08:40.39 44   1    256+  855+   58M+   56M+   12K    47M    299M+  1006M+ 248   1     sleeping 501
92740- adb_aos      0.0  00:00.47 5    0    83    62     484K   340K   0B     732K   40M    618M   92739 1     sleeping 501
92553  Console      0.0  00:07.40 3    0    158   221    4156K  2560K  0B     33M    36M    2509M  92553 177   sleeping 501
89792  com.apple.We 0.0  05:43.72 8    3    372   1057   54M    34M    0B     20M    67M    3556M  89792 1     sleeping 501
89788  Safari       0.0  16:47.59 13   1    2530  2811   134M   82M    812K   48M    505M   4237M  89788 177   sleeping 501
89632- dbfseventsd  0.0  00:06.00 1    0    7     27     4172K  4140K  0B     96K    4148K  591M   248   89631 sleeping 0
89631- dbfseventsd  0.0  00:02.02 1    0    14    26     40K    20K    0B     136K   5280K  583M   248   1     sleeping 0
88379  com.apple.sb 0.0  00:00.02 2    0    49    42     544K   368K   0B     384K   45M    2433M  88379 177   sleeping 501
83267  helpd        0.0  00:01.07 2    0    47    46     472K   304K   0B     696K   45M    2434M  83267 177   sleeping 501
82077  usbmuxd      0.0  00:01.18 3    0    44    45     384K   256K   0B     828K   55M    2423M  82077 1     sleeping 213
82038  iTunesHelper 0.0  00:02.75 2    0    65    70     952K   604K   0B     1628K  45M    2437M  82038 177   sleeping 501
78815  spindump_age 0.0  00:00.01 2    1    45    54     468K   288K   0B     596K   45M    2414M  78815 177   sleeping 501
```

Process Management

You can put a process as background with & (ampersand) after a command. A background job will keep running until it finishes. This allows users to work on different tasks while the background job running. Don't forget your background jobs are consuming resources (CPU, Memory, File I/O etc).

sleep	delay for a specified amount of time
&	run a process as a background job
Ctrl-z/fg	send a foreground job to background and vice versa

```
$ sleep 2
$ sleep 10 &
[1] 18506
$ fg
sleep 10
^Z
[1]+  Stopped    sleep 10
```

```
$ bg
[1]+ sleep 10 &
```

Process Management

ps	display currently running jobs
kill/killall	terminate a process (not for PBS job)

```
$ sleep 10 &
[1] 27629
$ ps
  PID TTY          TIME CMD
27362 pts/27   00:00:00 bash
27629 pts/27   00:00:00 sleep
27791 pts/27   00:00:00 ps
$ kill 27629
$ ps
  PID TTY          TIME CMD
27362 pts/27   00:00:00 bash
28248 pts/27   00:00:00 ps
[1]+  Terminated              sleep 10
```

Command-line Completion

Tab key shows candidates of command/file/directory names, or complete the rest of name automatically (tab completion). It can substantially reduce the number of keystrokes.

Extremely handy!!!

```
$ env > very-very-long-file-name.txt
$ cat very-            <- press tab key here
$ cat very-very-long-file-name.txt          <- completed
$ mkdir -p sub/{dirA,dir@,dir9}
$ ls sub/dir           <- press tab key here
dir@/ dir9/ dirA/      <- show candidates
$ ls sub/dirA
```

http://en.wikipedia.org/wiki/Command-line_completion

Other Special Characters

~	home directory
.	current directory
..	parent directory
*	any number of any character (wild card)

```
$ cd ~
$ pwd
/home/avalonjo
$ cd ..
$ pwd
/home
$ cd .
$ pwd
/home
$ cd ~
$ pwd
/home/avalonjo
```

```
$ cd workshop/
$ ls
emptyfile1  emptyfile3_withNewName
output_of_ls
emptyfile2  hello.sh
subDirectoryLevel1/
$ ls emp*
emptyfile1  emptyfile2
emptyfile3_withNewName
```

Command History

Shell keeps your command history. Always a good idea to review it if you forgot what to type. A special character **!** reruns a command in the command history.

history	Display command history
↑↓	Display command history
!	Rerun a command

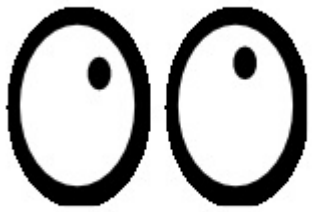
```
$ history
...
1030 ps axuw | grep -v ${USER} | sort -r -n -k 3 | head -n 5
1031 env > very-very-long-file-name.txt
1032 cat very-very-long-file-name.txt
1033 mkdir -p sub/{dirA,dirB}
1034 ls sub/dirA
$!1030          <- rerun the 1030th command
ps axuw | grep -v ${USER} | sort -r -n -k 3 | head -n 5
```

X

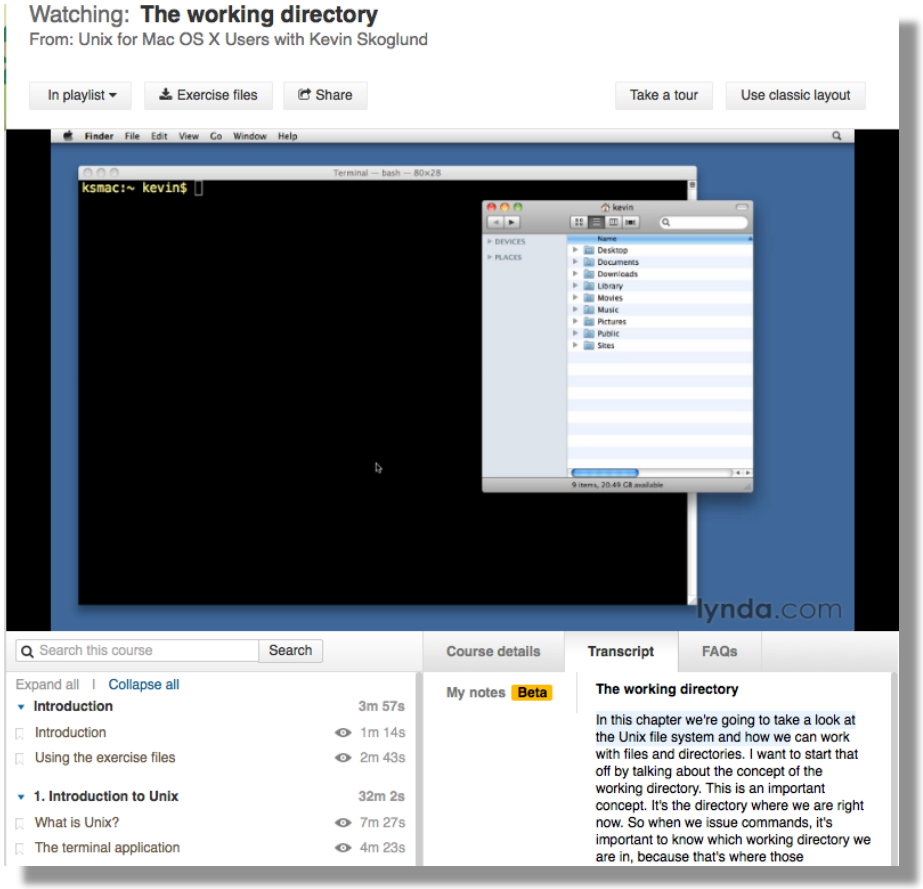
An X server is a program that displays graphics on your monitor. An X-client is a program that sends graphics data to your X-server so that it can be displayed on your monitor. XWin32 & mobaXterm are two Window's X clients.

ssh -X	Connect to a remote host with the intent of sending graphics data back to your workstation's monitor. Only works on Linux/Mac workstations.
XWin32	windows software that allows the same functionality

$ ssh –X hpc-login2.usc.edu
$ xeyes



Want to learn more?



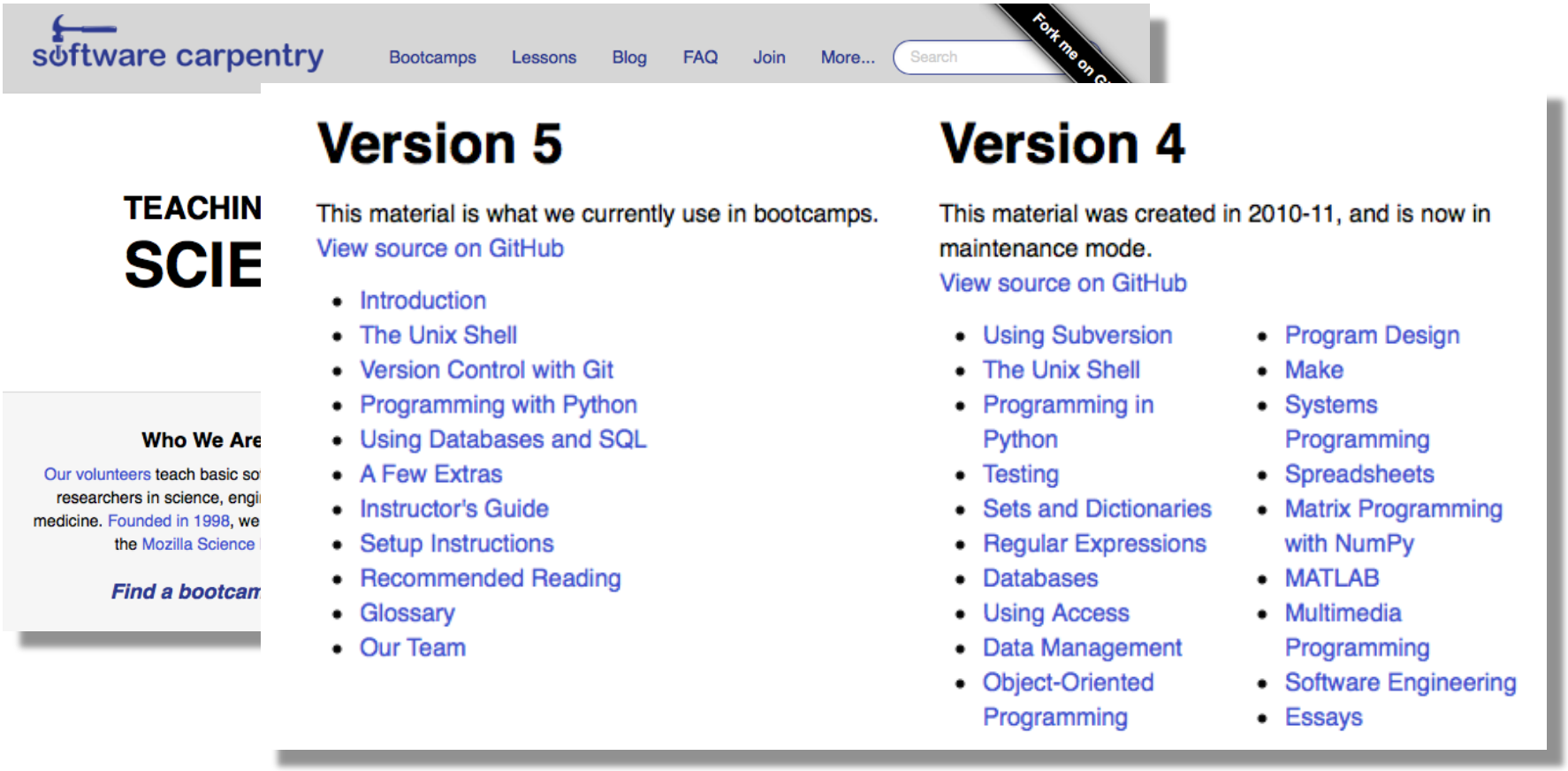
- Up and Running with Bash Scripting
- Unix for Mac OS X Users
- Using Regular Expressions
- Perl 5 Essential Training
- R Statistics Essential Training
- C/C++ Essential Training
- Up and Running with Python
- Python 3 Essential Training
- Up and Running with MATLAB
- Up and Running with R

and More!

From: Unix for Mac OS X Users

lynda.com

Want to learn more?



http://software-carpentry.org

A special thanks to:

Dr. Ken-Ichi Nomura Ph.D.

for slides upon which this presentation is based

Thank you!