

Brief Overview

This tutorial serves as an introduction to using the Center's large Dell Linux cluster. Topics of discussion include login, data transfer, and overview of basic Linux/Unix commands. The user environment, code compilation and basic performance analysis are highlighted. The tutorial closes how to get help and using the online resources.

Computational Resources:

The [U2](#) cluster comprises the major part of computational resources in CCR.

- The login machine and all compute machines run the RedHat Linux operating system.
- This cluster has 1056 dual processor compute machines (nodes).
 - Intel 64EMT processors
- All compute nodes are connected with Gigabit ethernet.
 - 768 nodes are also connected with Myrinet, a high speed fibre network.
- Several types of data storage are available, including 25 TB of shared network storage.

Software Resources:

CCR provides various [compilers](#), [Message Passing Interface](#) (MPI) implementations, and a wide variety of [application software](#).

- Many of the applications are available through the [web portals](#).

CCR Accounts

UB faculty interested in using CCR resources can apply for accounts by filling out an [online form](#).

Getting help

The [CCR web site](#) provides extensive online information, including tutorials and workshops. Users can [request assistance](#) and [consultation](#) on research projects.

CCR Web : Getting Started

This page last changed on Jan 27, 2008 by [furlani](#).



These guides provide detailed information for new and experienced users. Whether just starting out or an experienced user, there is important and current information (machine status, resource utilization (CPU cycle, storage, etc)) to assist you in effectively utilizing CCR resources.

Also, be sure to check out [CCR's new Web Portal offerings](#). CCR currently provides portals to support research in chemistry and bioinformatics, with additional portals under development. Portals are desirable because they provide convenient, easy-to-use web-based interfaces, thereby allowing scientists to focus on their research as opposed to details of the center's computing environment.

New Users

- [Getting an Account](#)
- [Introduction to CCR Web Portals](#)
- [Unix Commands](#)
- [User Environment](#)
- [Login and Logout](#)
- [Changing Password](#)
- [X-Display](#)
- [Transferring Data](#)
- [Home Directory and Quota](#)
- [Software](#)
- [Basic Compilation](#)
- [Running Jobs](#)
- [Basic Analysis and Monitoring](#)
- [Getting Help](#)
- [New Users Reference Card](#)
- [Frequently Asked Questions](#)

Experienced Users

- [Machine Status](#)
- [Current Usage Information](#)
- [Managing CCR Accounts](#)
- [Data Storage Resources](#)
- [Batch Computing](#)
- [Job Priority and Fairshare](#)
- [Compilers](#)
- [MPI and Parallel Computing](#)
- [Mathematics Libraries](#)
- [Scientific Visualization](#)
- [Debugging](#)
- [Profiling](#)
- [Application Software By Subject](#)
- [iNquiry Web Portal](#)
- [SeqWeb Web Portal](#)
- [WebMO Web Portal](#)
- [HPC Consulting](#)

This page last changed on Feb 13, 2008 by [cdc](#).

Linux Operating System

The [U2](#) cluster, as well as the login (front-end) machine run the Redhat Linux operating system. Linux is a variation of UNIX.

Command Line Interface

The user interface is a command line. There is no graphical user interface (GUI) by default, however **Linux supports the use of graphical applications**. Many scientific applications available on U2 have graphical interfaces.

Here is an example of the command line interface upon login:

```
----Latest News -----
```

```
August 7, 2007
```

```
The intel 10.0 and pgi 7.0 compilers are available.
```

```
The default for "module load intel" are the intel 9.1 compilers.  
The default for "module load pgi" are the pgi 6.2 compilers.
```

```
Older modules have been cleaned up. Please e-mail ccr-help@ccr.buffalo.edu  
if you need one that is no longer available.
```

```
----Monthly Downtime -----
```

```
We will have monthly downtime on the U2 cluster on the first  
Tuesday of the month starting at 7am. The cluster will be  
returned to production as soon as possible.
```

```
** Next Scheduled downtime is March 4th 2008 at 7:00am **
```

```
-----  
[bono:~]$
```

There are a few commands that will get users started working in the UNIX environment. These commands will be introduced in this tutorial.

Notes for Windows Users

As we proceed, special notes for Windows users will be highlighted.

Web Portals

The [Web Portals](#) provide convenient and easy-to-use interfaces to many of the command line scientific applications.

CCR Web : Login to U2

This page last changed on Jan 31, 2008 by [cdc](#).

Login

- Only Secure Shell (ssh) version 2 can be used to login to CCR machines.
 - Secure Shell is a network protocol that creates an encrypted channel between two computers.
- Users login to `u2.ccr.buffalo.edu`. The actual name of the front-end machine is `bono`.
- [u2 cluster details](#)
- The U2 front-end is accessible from machines in the UB domain (`buffalo.edu`).
- Your login username will be the same as your UBIT username.
- The initial password will be sent to you.

Login from Linux/Unix Machine

- `ssh u2.ccr.buffalo.edu`
 - Your machine must be on the UB network.
 - Use VPN from home or offsite: [UB-VPN](#)

Login from Windows Machine

- Secure Shell Client: [PuTTY](#) or [X-win32](#)
 - Your machine must be on the UB network.
 - Use VPN from home or offsite: [UB-VPN](#)

Logout

- On the command line, type **logout** or **exit**.

This page last changed on Jan 25, 2008 by [cdc](#).

Changing your password

- Login to the u2 front-end.
- Use the kpasswd command to change the password.
 - Type: kpasswd
- It takes a few minutes for the cluster to be updated.

This page last changed on Feb 19, 2008 by [cdc](#).

CCR's computing resources are primarily Linux based and therefore using them requires a basic understanding of the Unix operating system. Some basic commands are provided below.

Basic Unix Commands

- Show pathname of current directory: `pwd`
- List files: `ls`
- Make a directory: `mkdir directory-name`
- Change directory: `cd directory-name`
 - Change directory back to home directory: `cd`
- Copy a file: `cp old-filename new-filename`
- View a file:
 - `cat filename`
 - `more filename`
 - `less filename`
- Edit a file:
 - `emacs filename`
 - `vi filename`
- Delete a file: `rm filename`
 - Delete a directory (recursively): `rm -R directory-name`
 - All files and subdirectories are deleted
- Move a file: `mv old-filename new-filename`
- Change permissions:
 - Arguments to `chmod` command: `ugo+-rwx`
 - where `ugo` are user, group and other; `rwx` are read, write and execute
 - Add execute permission for yourself: `chmod u+x filename`
 - Remove read, write and execute for group and other from a directory its contents:
`chmod -R go-rwx directory-name`
- More Information:
 - [Unix Reference Card](#)
 - [emacs Reference Card](#)
 - [vi Reference Card](#)

Login Shell and Path

The U2 cluster runs the Linux operating system, which is a version of UNIX. The user interface is command line. When a user logs in to the front-end of the cluster, a login shell starts. This is the traditional command line user interface. Users input [UNIX commands](#) to the login shell.

The PATH environmental variable is a list of directories that is searched for a command. In order to use a command, either it must be in your path or you must specify the complete path.

Default Login Shell

- The default login shell for all users is bash ([GNU Bourne-Again Shell](#)).
 - View man page for bash: `man bash`
Users can request a different login shell; see [Getting Help](#).
- Show default shell: `echo $SHELL`

Default PATH

- Show the default path: `echo $PATH`
- Modifying the PATH.
 - Use [modules](#) to modify your PATH.
 - Add to the default path (bash): `export PATH=newpath:${PATH}`

Environmental Variables

- Show the environment (bash): `env`

[Modules Software Interface](#)

Introduction to Modules

Modules are used to modify paths and set variables for application packages.

- This is a convenient mechanism for managing your user environment.
- Modules prevents conflicting entries and mangling in the paths.
 - Loading a module will add application paths to PATH variables and set environmental variables necessary for the application.
 - Unloading a module will remove the the application specified entries from your PATH.
- Modules can use used on the command line and in PBS scripts (batch).

Commands

- **module avail**
 - Display a listing of available modules.
 - Example:

```
[bono:~]$ module avail
----- /util/Modules/versions -----
3.1.6
----- /util/Modules/3.1.6/modulefiles -----
adf/2005.01b                mpi-hmmer/v2.3.2-0.9
adf/2006.01b(default)      mpiblast/ch_p4/current
```

```

adf/2006.01d          mpich/gcc-3.4.6/ch_mx/1.2.7..4
adf/2007.01          mpich/gcc-3.4.6/ch_p4/1.2.7p1
amber/8              mpich/intel-10.0/ch_mx/1.2.7..4
amber/9(default)    mpich/intel-10.0/ch_p4/1.2.7p1
...

```

- **module avail application-name**

- List the modules available for a particular application.
- Example:

```

[bono:~]$ module avail adf
----- /util/Modules/3.1.6/modulefiles -----
adf/2005.01b          adf/2006.01d
adf/2006.01b(default) adf/2007.01
[bono:~]$

```

- **module load modulefile**

- Load the modulefile into your environment.
- The default module will be loaded if the full module name is not used.
- Example:

```

[bono:~]$ echo $PATH
/util/Modules/3.1.6/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/kerberos/bin:
/usr/local/maui/sbin:/usr/local/maui/bin:/usr/local/mx/bin:/usr/local/mx/sbin:
/san/user/user1/u2/bin
[bono:~]$ module load adf
'mpich/intel-9/ch_p4/1.2.7p1' load complete.
'adf/2006.01b' load complete.
[bono:~]$ echo $PATH
/util/java/jdk1.5.0_06/bin:/util/intel/cce/9.1.051/bin:/util/intel/fce/9.1.051/bin:/util/intel/
idbe/9.1.051/bin:/util/intel/eclipsepackage/3.0.1/eclipse:/util/mpich/1.2.7p1/intel-9/ch_p4/bin:/
util/mpich/1.2.7p1/intel-9/ch_p4/sbin:/util/adf/v2006.01b/bin:/util/Modules/3.1.6/bin:/usr/local/
bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/kerberos/bin:/usr/local/maui/sbin:/usr/local/maui/bin:/usr/
local/mx/bin:/usr/local/mx/sbin:/san/user/user1/u2/bin
[bono:~]$

```

- **module unload modulefile**

- Unload the modulefile from your environment.
- Example:

```

[bono:~]$ module unload adf
'mpich/intel-9/ch_p4/1.2.7p1' unload complete.
'adf/2006.01b' unload complete.
[bono:~]$ echo $PATH
/util/Modules/3.1.6/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/kerberos/bin:/usr/local/
maui/sbin:/usr/local/maui/bin:/usr/local/mx/bin:/usr/local/mx/sbin:/san/user/user1/u2/bin
[bono:~]$

```

- **module list**

- Shows the modules currently in your environment.
- Example:

```

[bono:~]$ module list
Currently Loaded Modulefiles:
1) null                    5) intel/9.1
2) modules                 6) mpich/intel-9/ch_p4/1.2.7p1
3) use.own                 7) adf/2006.01b
4) java/j2sdk/1.5.0_06
[bono:~]$

```

- **module show modulefile**

- Shows the paths and variables set by a module.
- Example:

```

[bono:~]$ module show adf
-----
/util/Modules/3.1.6/modulefiles/adf/2006.01b:

```

```

module-whatIs    ADF Quantum Chemistry Package
setenv           ADFHOME /util/adf/v2006.01b
setenv           ADFBIN /util/adf/v2006.01b/bin
setenv           ADFRESOURCES /util/adf/v2006.01b/atomicdata
setenv           SCMLICENSE /util/adf/v2005.01b/license
setenv           MPIDIR /util/mpich/1.2.7p1/intel-9/ch_p4
prepend-path     PATH /util/adf/v2006.01b/bin
setenv           NSCM 1
module           load mpich/intel-9/ch_p4/1.2.7p1
-----
[bono:~]$

```

- **module help modulefile**
 - Display extended help listing of modulefile, often contains detailed links to further informational resources.

Using modules in batch

- If you change shells in your batch script you may need to explicitly load the modules environment (if your script does not use modules commands, there is, of course, no need to do this) by adding a the following line (e.g. for bash) to your bash script:

```
. ${MODULESHOME}/init/bash
```

- More examples are available in [Sample PBS scripts](#).

Creating your own modulefiles

- You can create your own modules files, for example, in organizing your own software installs and development trees.
- The module **use.own**, when loaded, allows modules to look at your **\$HOME/privatemodules** directory for additional modules that you can then access through the modules interface.
- On CCR's systems, you can refer to the **\$MODULESHOME/modulefiles/template** for a reference modulefile.
 - Also see the modulefile man page for a detailed description of the syntax used in these files.

Other sources of information on modules

- The [module man page](#).
- The [modulefile man page](#) (useful syntax reference when writing your own module files).
- The [modules sourceforge page](#).
- Modules documentation at [NERSC](#).

Data Files from Windows Workstations

Remove Windows control characters from text files.

- `dos2unix -n oldfile.txt newfile.txt`

Spaces embedded in filenames and directory names

- Use `"\"` followed by a space to specify filenames with spaces.
- Convert the embedded spaces to underscore with the `convertspaces.zsh` script.

This page last changed on Feb 19, 2008 by [cdc](#).

The X Window System is a display protocol which provides windowing or graphical user interfaces (GUIs) on most Unix-like operating systems, including those maintained by CCR. Included here are instructions for running an X-window environment on U2, CCR's production supercomputer.

X-Display from u2

X-Display to Linux/Unix Machine

- `ssh -X u2.ccr.buffalo.edu`
 - In most case the `-X` flag will work. If this fails, then try the `-Y` flag.
 - Your machine must be on the UB network.
 - Use VPN from home or offsite: [UB-VPN](#)

X-Display to Windows Machine

- The [X-win32](#) must be started before the [PuTTY](#) Client.
- In the PuTTY client enable X11 forwarding.
 - Click on X11 under SSH in the left column.
 - Click on box to enable X11 forwarding.
 - Click on Open to connect.
- Your machine must be on the UB network.
- Use VPN from home or offsite: [UB-VPN](#)

This page last changed on Jan 25, 2008 by [cdc](#).

File Transfer from Linux/Unix Machine

Secure File Transfer

- sftp u2.ccr.buffalo.edu
 - Upload to u2: put filename
 - Download from u2: get filename

Secure Copy

- scp filename u2.ccr.buffalo.edu:filename

File Transfer from Windows Machine

- [WinSCP](#) provides a drag and drop interface.

CCR Web : Home Directory and Quota

This page last changed on Jan 25, 2008 by [cdc](#).

User disk space at CCR is contained in a high-performance Storage Area Network (SAN). CCR's [SAN](#) provides a total of 20TB of disk storage to the U2 cluster. Directories are NFS mounted to the U2 cluster and accessible from all compute nodes.

User Home Directories

- Home directories on the SAN in the /san/user filesystem.
 - The home directory for the U2 cluster has the form:
 - /san/user/UBIT-username/u2
 - Data in the home directories are backed up to a tape library.
 - [CCR Backup and Retention Policy](#)
 - Home directories are accessible from all U2 compute nodes.

User Quota

- The default quota for a home directory is 2 Gigabytes.
- Use the quota command to display current usage and quota limit.
 - Type: `quota -u UBIT-username`

SAN

User disk space at CCR is contained in a high-performance Storage Area Network (SAN). CCR's [SAN](#) provides a total of 20TB of disk storage to the [U2](#) cluster. Directories are NFS mounted to the U2 cluster and accessible from all compute nodes.

- Home directories: /san/user
 - Example: /san/user/UBIT-username/u2
 - Default user quota is 2GB.
 - Backed up to a tape library according to the [CCR Backup and Retention Policy](#).
- Projects directories: /san/projects1, /san/projects2 and /san/projects3
 - Additional disk space is available for research groups in the project directories.
 - Faculty interested in project disk space should [contact the CCR staff](#).
 - The default group quota is 100GB.
 - By default, there is **NO** backup of data in the group project directories.
- Global scratch space (2TB): /san/scratch
 - Available to all users for temporary use.
 - There is **NO** backup of these data.

IBRIX

The [IBRIX](#) file system provides 25TB of high performance global scratch space. IBRIX is accessible to users while running applications. This high performance storage is recommended for applications that require significant I/O throughput. Please note that it is volatile storage and not meant to serve as permanent storage.

- /ibrix/scratch
 - Accessible from all compute node in the U2 cluster.
 - Available to all users for temporary use.
 - There is **NO** backup of these data.

Local Scratch Space

All compute nodes in the U2 cluster have local disk space (/scratch).

- This scratch space is available to [batch jobs](#) running on the cluster.
 - The variable \$PBSTMPDIR is set to /scratch/\$PBS_JOBID.

[Overview of U2 Disk Layout](#)

This page last changed on Feb 13, 2008 by cdc.

Code Compilation

Using Application Software

CCR offers a wide range of [scientific application software](#). These programs are already compiled for the [U2](#) and ready to use.

Users interested in running freeware, shareware, or commercial software not currently available on [U2](#) should [contact the CCR staff](#). Typically, CCR staff will install requested software in the /util directory.

Additionally, users may install software in [home and project directories](#). [Help](#) with installing and verifying functionality is available.

Types of Compilers

Various Fortran and C/C++ compilers are available. Currently, CCR provides compilers from [GNU](#), [INTEL](#), and [PGI](#).

Serial and Parallel Codes

Serial codes are programs that run as a single process. These codes are typically compiled with one of the C/C++ or Fortran compilers.

Parallel codes are programs that run as several processes on the same machine or different machines.

Some compilers have flags that enable automatic parallelization of the code. The resulting executable can utilize more than one process on a single machine, but often the speedup is not significant, except for the simplest of codes.

Running over many machines requires some form of message passing, the most popular of which is [Message Passing Interface \(MPI\)](#). These codes are compiled with wrapper scripts that enable linking to the MPI libraries. Wrapper scripts are available for the GNU, INTEL, and PGI compilers.

CCR Web : Basic Compilation

This page last changed on Feb 13, 2008 by [cdc](#).

This guide is an introduction to compiling C and Fortran programs. The samples are compiled with the [GNU](#) compilers. The [Intel and PGI compilers](#) are available.

Using the GNU Compilers

The GNU compilers are in the default path.

GNU C Compiler

- Code: Hello World

```
[bono:~]$ cat hello.c
#include <stdio.h>
main()
{
printf("Hello World!\n");
}
[bono:~]$
```

- Compilation:

```
[bono:~]$ gcc -o hello-gnu-c hello.c
```

- Running the code:

```
[bono:~]$ ./hello-gnu-c
Hello World!
[bono:~]$
```

GNU Fortran Compiler

- Code: Hello World

```
[bono:~]$ cat hello.f
C      Hello World (Fortran 77)
PROGRAM HELLO
PRINT*, 'Hello World!'
END
[bono:~]$
```

- Compilation:

```
[bono:~]$ g77 -o hello-gnu-f hello.f
```

- Running the code:

```
[bono:~]$ ./hello-gnu-f
Hello World!
[bono:~]$
```

[Using MPI with GNU Compilers](#)

CCR Web : Introduction to Batch Computing

This page last changed on Feb 15, 2008 by [cdc](#).

Overview

In general, jobs are submitted to CCR resources using a batch queuing system.

- This insure an equitable distribution of computing resources among all users.
- Each submitted job is assigned a priority, which determines when the job will run.
 - Details of the priority scheme are available at [CCR Job Priority](#).

PBS

The scheduler used in CCR is an open-source version of PBS (Portable Batch Scheduler).

Execution Model

- User submits a request to the scheduler.
 - This request is usually a script.
- The scheduler assigns compute nodes to the job.

[Schematic \(pdf\)](#)

Command Line Tools

top

- The **top** command actively monitors processes on a machine.
 - Displays CPU and memory utilization.
 - Processes consuming CPU and memory resources.
 - **PID** is the process ID of a task.
 - Example of running top on the [U2](#) front-end machine.

```
top - 11:43:01 up 8 days, 3:29, 66 users, load average: 0.44, 0.38, 0.87
Tasks: 407 total, 1 running, 401 sleeping, 3 stopped, 2 zombie
Cpu(s): 5.0% us, 1.1% sy, 0.0% ni, 92.9% id, 0.9% wa, 0.0% hi, 0.2% si
Mem: 4045144k total, 2302052k used, 1743092k free, 5996k buffers
Swap: 8193140k total, 861336k used, 7331804k free, 488456k cached
```

PID	USER	PR	NI	%CPU	TIME+	%MEM	VIRT	RES	SHR	S	COMMAND
9514	root	16	0	9	206:47.32	0.7	35496	25m	1384	S	pbs_server
19770	user1	15	0	4	3:13.07	2.4	335m	96m	20m	S	firefox-bin
24126	user2	16	0	1	62:49.60	0.0	57600	1572	1000	S	sshd
16524	user3	16	0	1	0:00.04	0.0	5560	1344	816	R	top
2138	user4	16	0	0	0:15.33	0.1	57576	2328	1280	S	sshd
4495	root	16	0	0	223:20.70	0.4	66384	15m	816	S	fm_server
10493	user4	16	0	0	3:30.97	0.0	58008	1928	1200	S	sshd
15988	user5	15	0	0	0:00.04	0.0	53244	1516	1104	S	bash
1	root	15	0	0	0:02.87	0.0	4756	464	432	S	init
2	root	RT	0	0	0:03.22	0.0	0	0	0	S	migration/0
3	root	34	19	0	0:02.78	0.0	0	0	0	S	ksoftirqd/0
4	root	RT	0	0	0:02.61	0.0	0	0	0	S	migration/1
5	root	34	19	0	0:00.47	0.0	0	0	0	S	ksoftirqd/1
6	root	RT	0	0	0:03.18	0.0	0	0	0	S	migration/2
7	root	34	19	0	0:00.82	0.0	0	0	0	S	ksoftirqd/2
8	root	RT	0	0	0:02.87	0.0	0	0	0	S	migration/3
9	root	34	19	0	0:00.58	0.0	0	0	0	S	ksoftirqd/3

Using the Unix Pipe

- A pipe is used to allow the output of one command to be sent as input to another command.
 - The pipe is represented with | (shift backslash key)

grep

- The **grep** command searches for a pattern.
 - Example that counts occurrences of a text in a file:

```
[bono:~/pbs-examples]$ grep c06 nodelist
c06n19
c06n16
c06n02
[bono:~/pbs-examples]$ grep c06 nodelist | wc -l
3
[bono:~/pbs-examples]$
```

file

- The **file** command determines the file type.

```
[bono:~/pbs-examples]$ file cpi.c
cpi.c: ASCII C program text
[bono:~/pbs-examples]$ file cpi-intel-9.1
cpi-intel-9.1: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), for GNU/Linux 2.4.0,
dynamically linked (uses shared libs), not stripped
[bono:~/pbs-examples]$
```

Idd

- The **ldd** command will show the shared library dependencies.

```
[bono:~/pbs-examples]$ ldd cpi-intel-9.1
libpthread.so.0 => /lib64/tls/libpthread.so.0 (0x000000317dd00000)
librt.so.1 => /lib64/tls/librt.so.1 (0x000000317eb00000)
libm.so.6 => /lib64/tls/libm.so.6 (0x000000317d700000)
libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x000000317e900000)
libc.so.6 => /lib64/tls/libc.so.6 (0x000000317d400000)
libdl.so.2 => /lib64/libdl.so.2 (0x000000317d900000)
/lib64/ld-linux-x86-64.so.2 (0x000000317d200000)
[bono:~/pbs-examples]$
```

nm

- The **nm** command lists symbols from an object file
 - piped through more

```
[bono:~/pbs-examples]$ nm cpi-intel-9.1 | more
000000000440c70 r _2il0floatpacket.1
000000000440eb0 r _2il0floatpacket.1
000000000444470 r _2il0floatpacket.1
0000000004445d8 r _2il0floatpacket.1
000000000447ce8 r _2il0floatpacket.1
000000000448498 r _2il0floatpacket.1
00000000044a3f0 r _2il0floatpacket.1
000000000440c78 r _2il0floatpacket.2
000000000444480 r _2il0floatpacket.2
0000000004445e0 r _2il0floatpacket.2
00000000044a400 r _2il0floatpacket.2
000000000440c80 r _2il0floatpacket.3
000000000444490 r _2il0floatpacket.3
0000000004445e8 r _2il0floatpacket.3
00000000044a410 r _2il0floatpacket.3
000000000440c88 r _2il0floatpacket.4
00000000044a420 r _2il0floatpacket.4
000000000440c60 r _2il0floatpacket.5
000000000440c90 r _2il0floatpacket.6
000000000440c98 r _2il0floatpacket.7
00000000043f7ca t A0Q0
00000000043f7c6 t A0Q1
--More--
```

- **nm** used to display references to MPI

```
[bono:~/pbs-examples]$ nm cpi-intel-9.1 | grep -i mpi | more
000000000409d88 T MPI_Attr_get
00000000040a40c T MPI_Attr_put
00000000040b724 T MPI_Barrier
00000000040b800 T MPI_Bcast
0000000004290c8 T MPI_Cancel
000000000564a98 B MPICHX_QOS_BANDWIDTH
000000000564a9c B MPICHX_QOS_PARAMETERS
00000000040ab78 T MPI_Comm_free
00000000040adc0 T MPI_Comm_rank
00000000040ae64 T MPI_Comm_set_name
```

```

000000000040b034 T MPI_Comm_size
0000000000422a8e T MPID_Abort
0000000000426b02 T MPID_ArgSqueeze
0000000000433a2a T MPID_BsendContig
00000000004237c8 T MPID_BSwap_N_copy
00000000004238ce T MPID_BSwap_N_inplace
0000000000571088 B MPID_byte_order
0000000000425416 T MPID_ByteSwapInt
000000000042495c T MPID_Cancel_print_pkt
0000000000425012 T MPID_CH_Abort
0000000000438174 T MPID_CH_Check_incoming
00000000004252e4 T MPID_CH_Comm_msgrep
--More--

```

ps

- The **ps** command shows a snapshot of the current processes on the system
 - The **-ef** flags will show all processes in full display.
 - The use of a pipe and **grep** can be used to refine the output.

```

[bono:~/pbs-examples]$ ps -ef | grep user1
root      10651  4272  0 16:09 ?          00:00:00 sshd: user1 [priv]
user1    10723 10651  0 16:09 ?          00:00:00 sshd: user1@pts/34
user1    10785 10723  0 16:09 pts/34    00:00:00 -bash
user1    11334     1  0 16:09 pts/34    00:00:00 /bin/sh /usr/bin/clustervis -h
c23n20,c23n19,c23n18,c23n17,c23n16,c22n24,c22n06,c21n25
user1    11870 11334  0 16:09 pts/34    00:00:00 pmview -h c23n20 -title SGI PCP : Cluster Node
Activity -xrm *iconName: clustervis
user1    11877 11870  0 16:09 ?          00:00:00 /usr/share/pcp/bin/pmtime -h -p /tmp/pmview.agWVsT
-D 0
user1    12083 10785  0 16:10 pts/34    00:00:00 ps -ef
user1    12084 10785  0 16:10 pts/34    00:00:00 grep user1
[bono:~/pbs-examples]$

```

- The **-f** and **-u** username will display all processes belong to that user

```

[bono:~/pbs-examples]$ ps -f -u user1
UID      PID  PPID  C  STIME TTY          TIME CMD
user1    10723 10651  0 16:09 ?          00:00:00 sshd: user1@pts/34
user1    10785 10723  0 16:09 pts/34    00:00:00 -bash
user1    20863     1  0 16:43 pts/34    00:00:00 /bin/sh /usr/bin/clustervis -h c
user1    21382 20863  0 16:43 pts/34    00:00:00 pmview -h c23n20 -title SGI PCP
user1    21387 21382  0 16:43 ?          00:00:00 /usr/share/pcp/bin/pmtime -h -p
user1    21897 10785  0 16:44 pts/34    00:00:00 ps -f -u user1
[bono:~/pbs-examples]$

```

Graphical Tools

[Debugging](#)

[Profiling](#)

CCR Web : Summary Introduction to CCR

This page last changed on Feb 19, 2008 by [cdc](#).

Online Resources

- The CCR website provides extensive information and online training.
 - www.ccr.buffalo.edu
- The best place to start is the [Getting Started](#) page.
 - There is information available for both new and experience users.
- The [Training](#) page has a wide variety of online tutorials.
 - Tutorials are under active develop most of the time.
 - If you would like training or a new tutorial please [contact us](#).
- [Web Portals](#) are available.

Assistance to Users

- Users can [send e-mail to CCR staff](#)
- CCR offers scientific, visualization, and database management [Consulting Services](#).