COLORADO SCHOOL OF MINES
EARTH • ENERGY • ENVIRONMENT

High Performance Computing Group

http://hpc.mines.edu

Timothy H. Kaiser, Ph.D.
hpcinfo@mines.edu

http://geco.mines.edu/prototype/I_am_new_to_linux_Help/

http://geco.mines.edu/prototype/Show_me_some_local_HPC_tutorials/fall17/

Linux for HPC

Timothy H. Kaiser, Ph.D.
Spring 2014 - Fall 2017

Warning!

This presentation is like trying to drink from a firehose. It is very fast and there is a lot of content.

You may feel overwhelmed.

The suggested usage is to go through it once to get an idea of what is possible.

Then go back and look of things that might be of interest.

For more information see:
http://hpc.mines.edu

If you have questions please email us at
hpcinfo@mines.edu.

Linux for HPC

- Overview
- File system (more details next slide)
- Logging in
  - ssh
  - Some tools for doing ssh
  - What happens
- Environment
  - what it effects
  - how it gets set
  - how to change it
  - modules

Files Linux, more details

- The file system
  - moving around
  - listing
  - hidden files
  - “wildcards”
  - deleting
  - creating/removing directories
  - creating files
    - touch
    - edit
    - pipe
    - redirect

Feature	Example
Moving around in the file	cd
Listing files	ls
Where are we	pwd
Hidden files	.bashrc, .ssh
Wildcards	*,{a,b,c},[0-9]
Remove	rm, rmdir
Creating directories	mkdir

Editing

- gedit (GUI)
- gvim (GUI)
- emacs (GUI)
- nano
- Remote editing
- Not covered
  - vi
  - emacs

Creating programs

- compilers
- make
- configure
- cmake

Advanced ssh

- Setting up keys
- .ssh/config file
  - alias
  - Specific configurations
  - Tunneling
- External tunneling

HPC and Parallel Programming

- What is it?
- How do you run parallel

Not or briefly covered…

- Running HPC applications
- Shell programming very briefly covered…
  - See: http://geco.mines.edu/scripts/
  - This was recently updated updated to include changes in our current system
  - Skip everything between “Running Bash Scripts” and “Redirection”

Let's get going…

Operating Systems

- What is an operating system
  - What a user uses to use the computer
  - It is the interface between a user and the computer
    - Controls the computer
    - Presents “stuff” to the user
  - Interface
    - GUI
    - **Text**
    - Voice/Sound

From Unix to Linux

- The Unix operating system was conceived and implemented by Ken Thompson and Dennis Ritchie (both of AT&T Bell Laboratories) in 1969 and first released in 1970.
- The History of Linux began in 1991 with the commencement of a personal project by a Finnish student, Linus Torvalds, to create a new free operating system kernel.
- Linux has been ported to all major platforms and its open development model has led to an exemplary pace of development.
- http://en.wikipedia.org/wiki/History_of_Linux

Many different “versions”

- Linux
  - http://en.wikipedia.org/wiki/Linux_distribution
- Linux/Unix like:
  - Mach (OSX - IOS)
  - AIX
  - Unicos
  - HP-UX

Some links

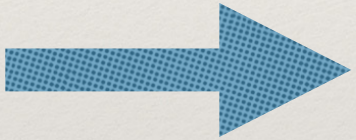
- Tutorials:
  - http://www.tutorialspoint.com/unix/index.htm
  - http://www.ee.surrey.ac.uk/Teaching/Unix/
  - https://www.cac.cornell.edu/VW/Linux/default.aspx?id=xup_guest
  - http://tille.garrels.be/training/bash/
  - See: http://geco.mines.edu/scripts/
- General Interest
  - http://en.wikipedia.org/wiki/History_of_Linux
  - http://en.wikipedia.org/wiki/Linux_distribution

Some cool things

- You learn Linux you will:
  - Be comfortable at just about any HPC site
  - Be better at working in OSX
  - Windows? - can help there also
- Built in help for most commands

Basic Interactions

- We will be talking mostly about text based interactions
- Even text based interactions need a terminal window
  - Linux - X11 based
  - OSX - Terminal
  - Windows



putty	BitVice	ZOC	VanDyke
bash	cygwyn	Firefox	MobaXterm

- Terminal programs
  - http://en.wikipedia.org/wiki/List_of_terminal_emulators
- Putty Instructions: http://geco.mines.edu/ssh/

ssh

- Command for getting on a Linux box from another
- Basic syntax from a terminal window:
  - ssh machine_name
    - ssh bluem.mines.edu
  - ssh username@machine_name
    - ssh tkaiser@bluem.mines.edu
  - ssh username@machine_name command
    - ssh tkaiser@bluem.mines.edu **ls**
  - To enable a remote machine to open a local window you need to add the -Y option

ssh -Y mio.mines.edu

ssh

- Reads a local configuration file ~/.ssh/config (if it exists)
  - Alias
  - Special password settings
  - Tunnels
- Sets up an encrypted connection between your local and remote machines
- “Normally” asks for a password (MultiPass)
- Opens up a session on the remote host in which you can enter commands
- Type exit to quit

A minor Digression

- You need to run some program on your machine to enable ssh
- On OS X and Linux boxes the software is preinstalled
- On Windows there are a number of options
- We'll look at some of them

Back to the ssh GUIs:

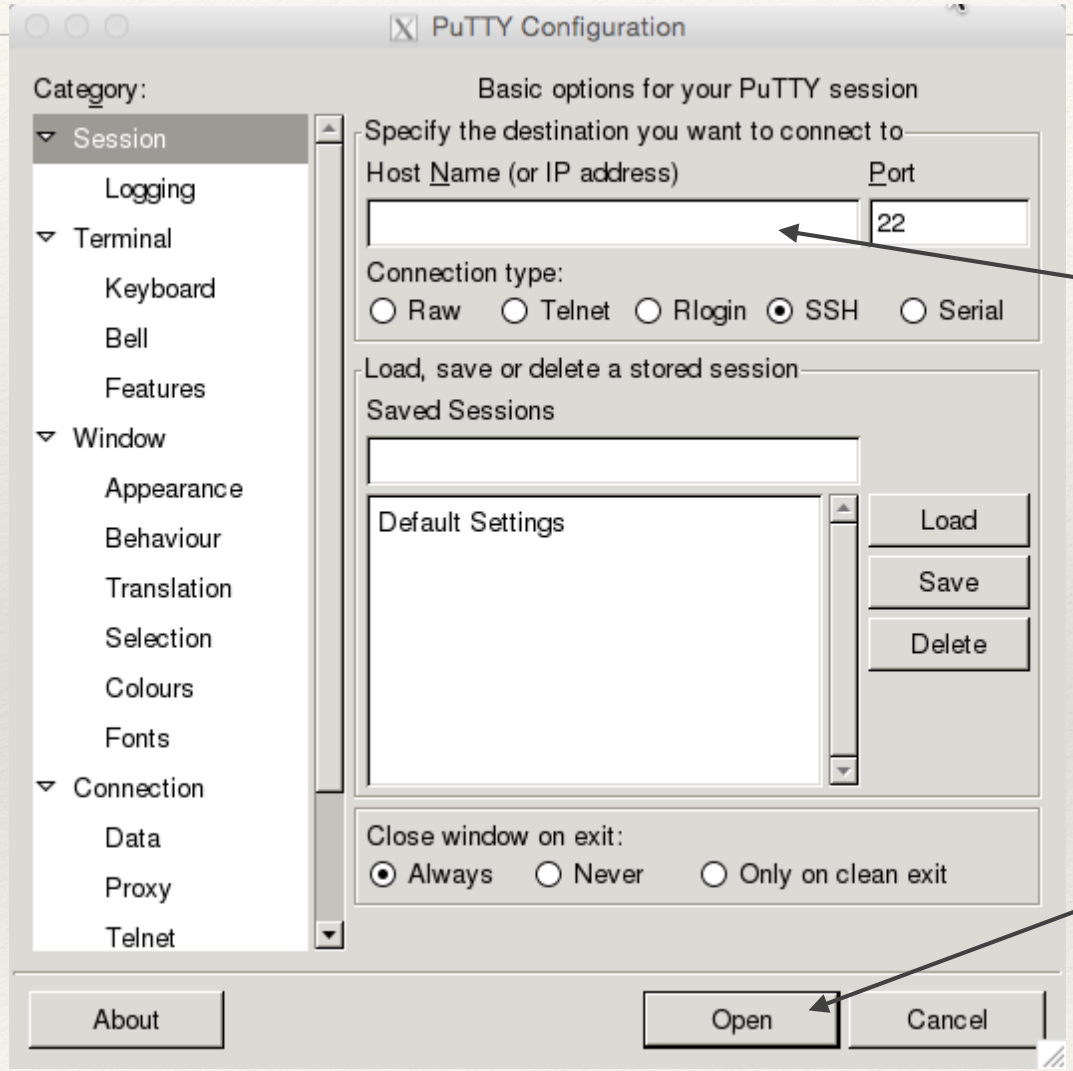
- There are a number of GUI clients that wrap ssh
  - Windows - putty
    - GUI for making connections
    - GUI for set up
    - Also provides a “normal” terminal window
    - Instructions
      - http://geco.mines.edu/ssh/puttyra.html
- Bitvise SSH Client
- Firefox plugin FireSSH and FireFTP (very cool)

These also provide “scp” for moving files between machines

Putty

- Common on Windows machines
- Easy to install
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  - You want “A Windows installer for everything except PuTTYtel"
- Allows an easy connection to Linux boxes

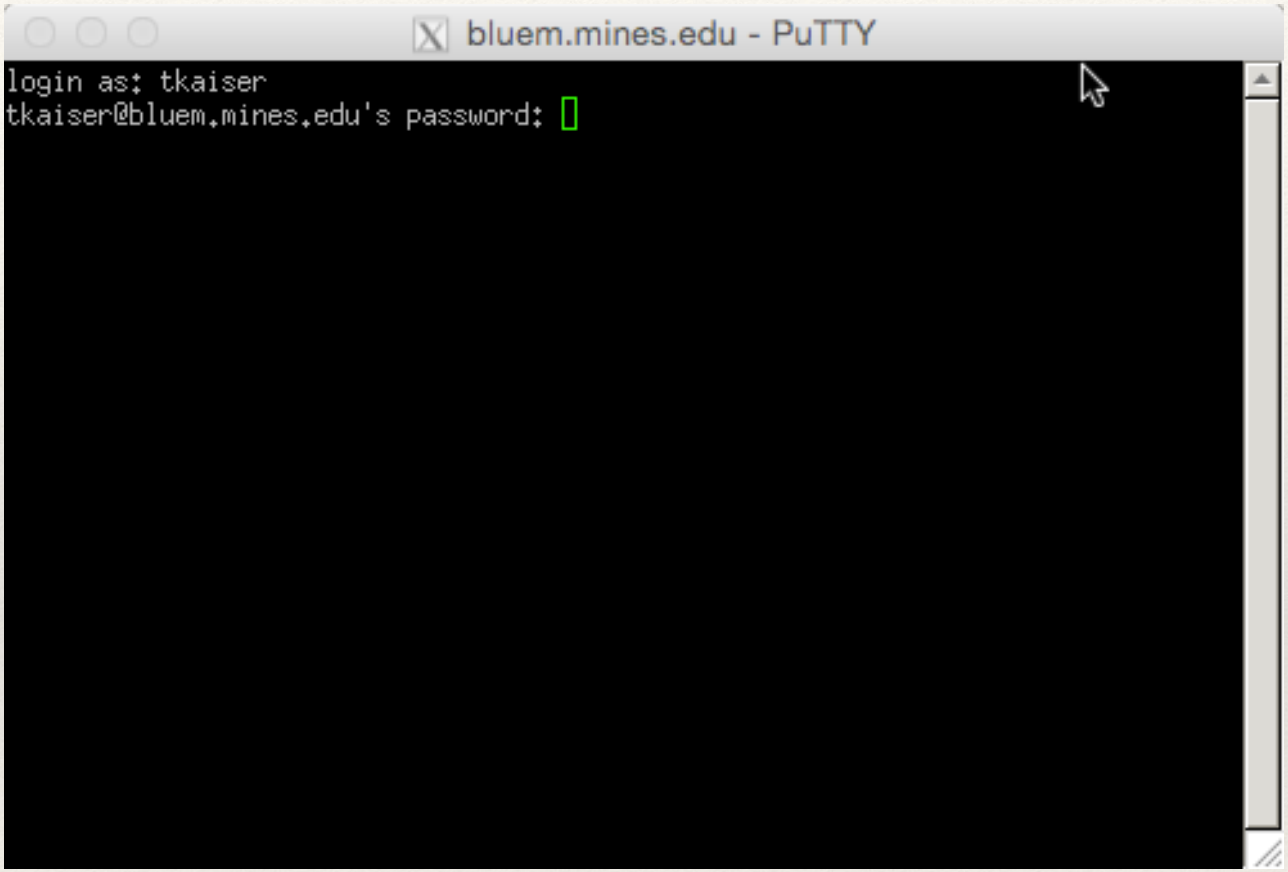
Putty setup Window



Enter the name of the machine to which you want to connect here.

Then click open

Putty Terminal Window

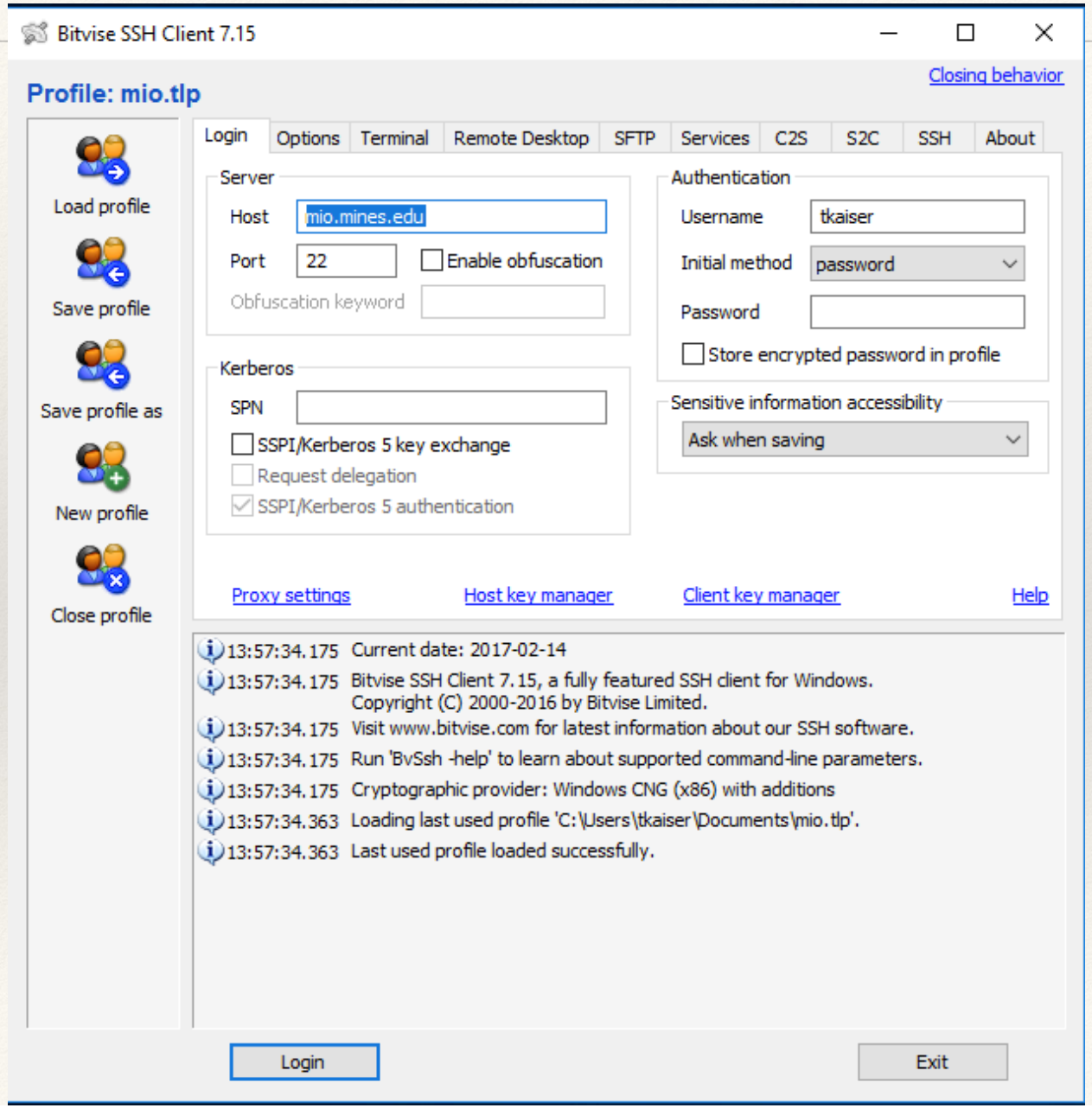


- Lots more options, see:
  - http://geco.mines.edu/ssh/puttyra.html

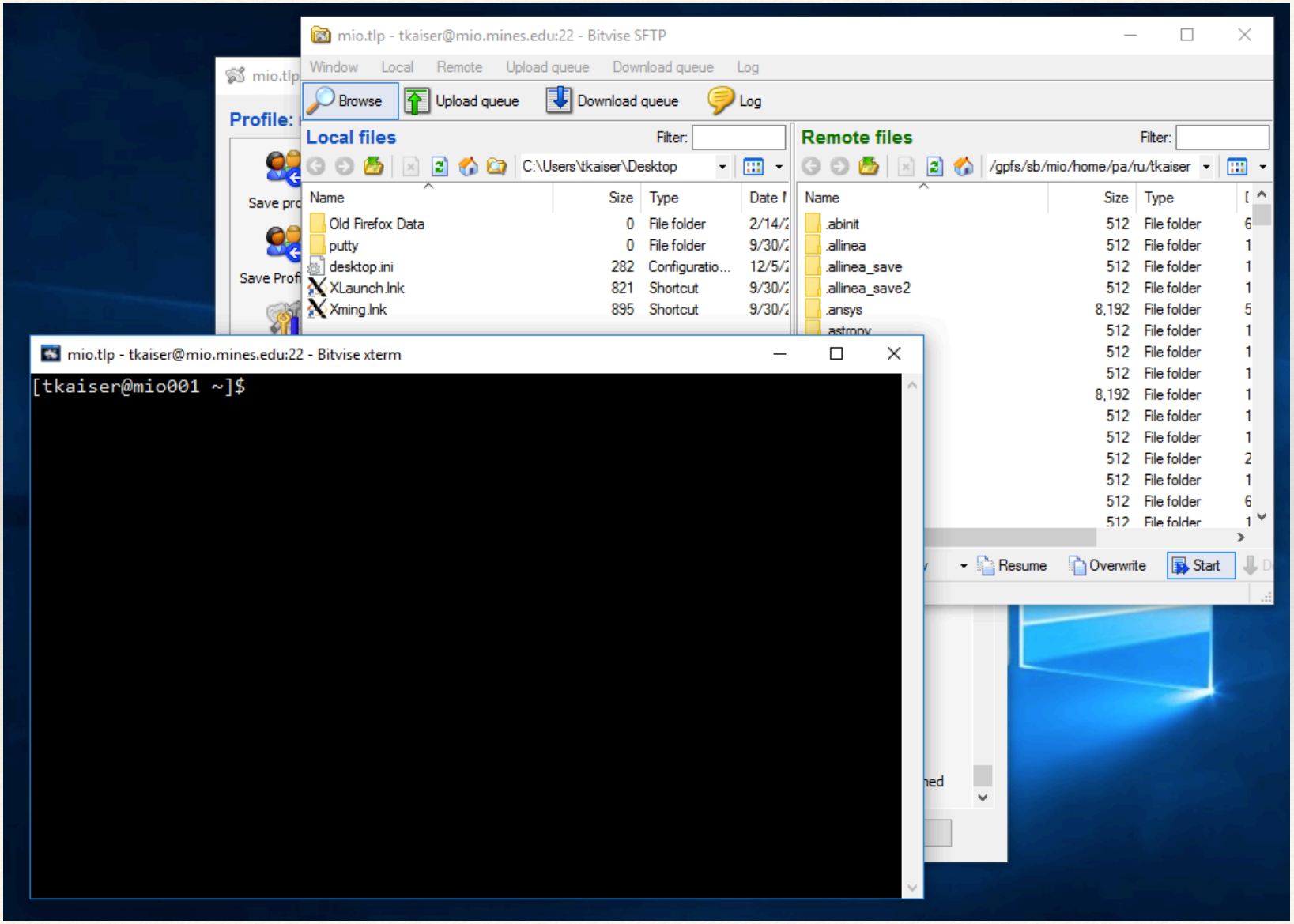
Bitvise SSH Client

- Bitvise SSH Client - like a commercial version of putty
- Free Client
- Windows Only
- https://www.bitvise.com/ssh-client
- Can open both terminal and transfer window

Bitvise SSH Client - Setup

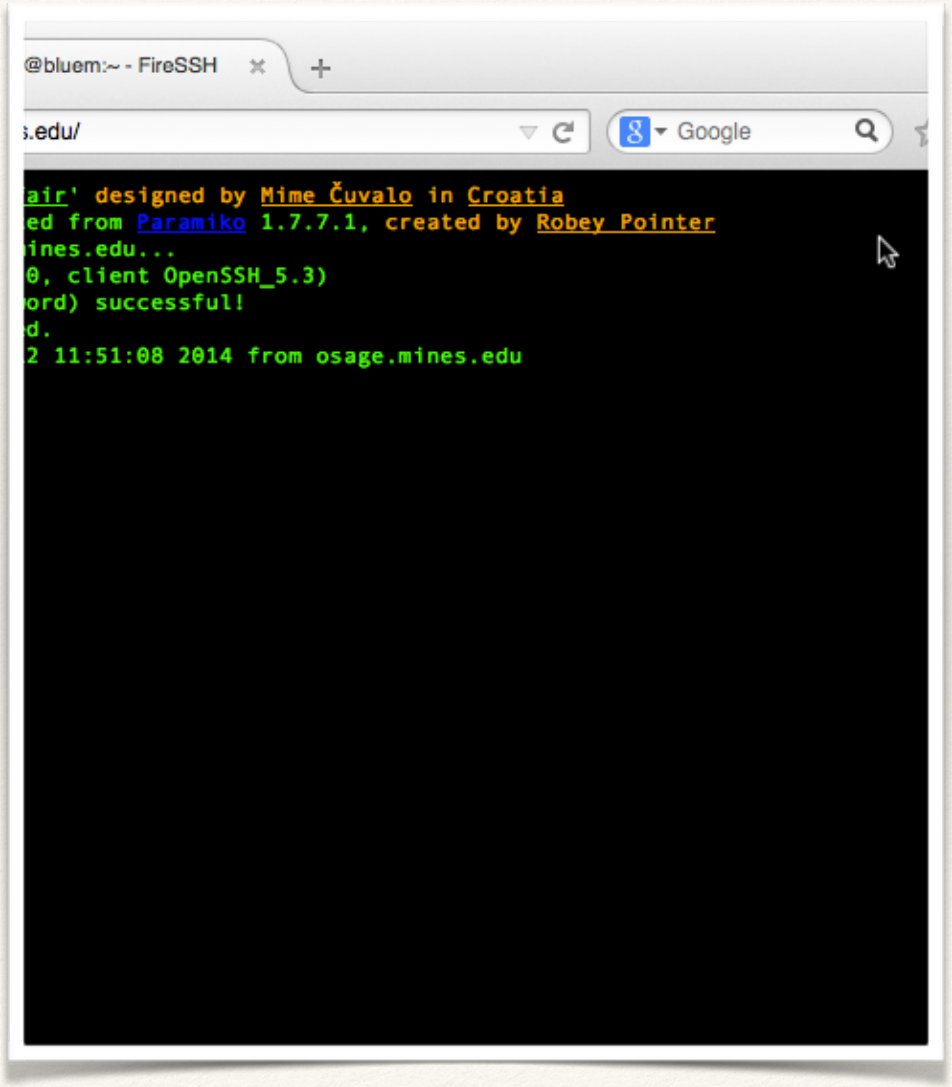


Bitvise SSH Client - Windows

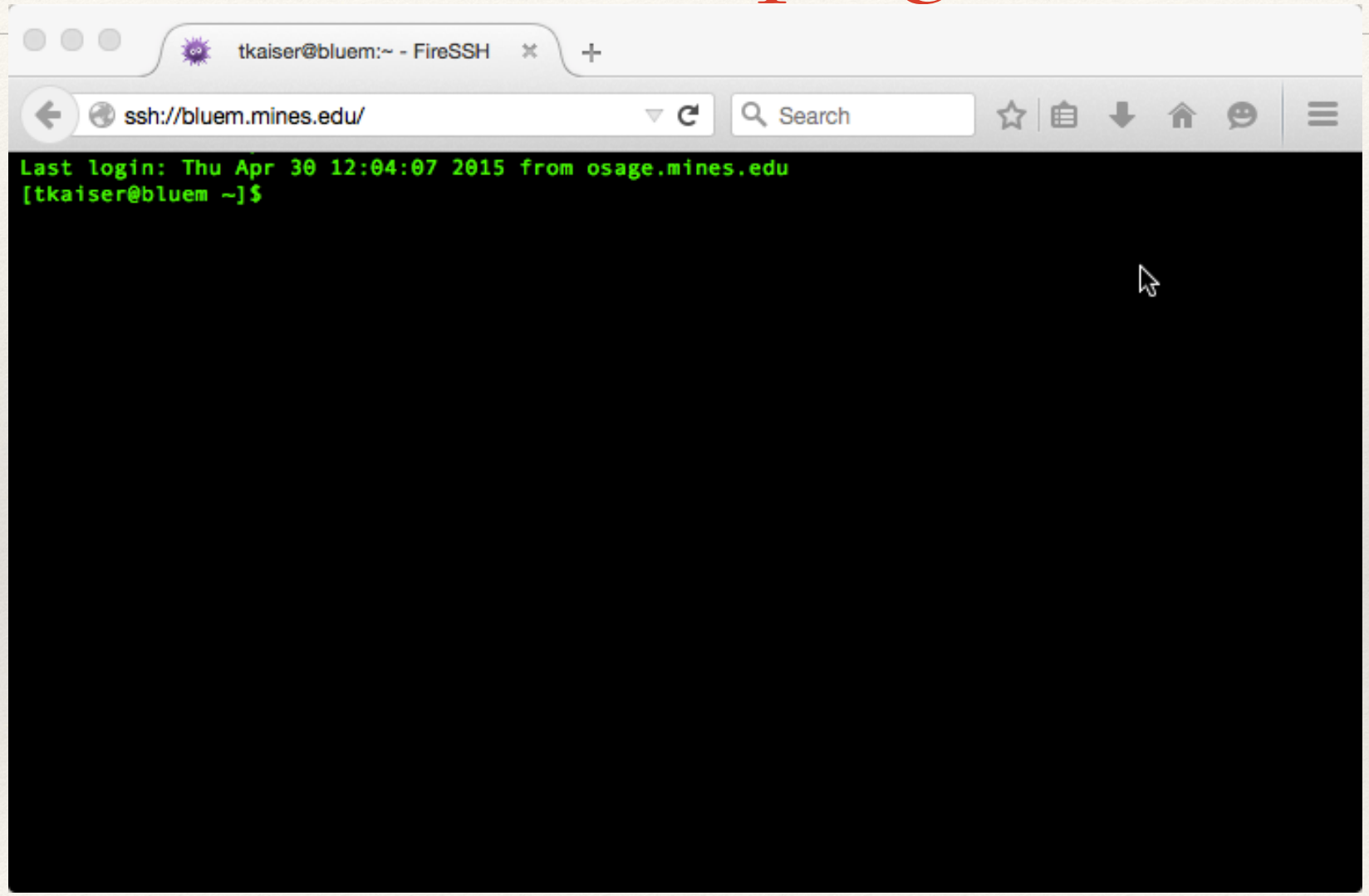


Firefox web based interactions

- There is a extension for the Firefox browser called **FireSSH** that gives you a terminal window within Firefox
- Very cool and works well!
- To launch it you just enter the name of the machine if the form ssh://bluem.mines.edu
- By the way…
  - If you try this in Safari it will open up the Terminal Application



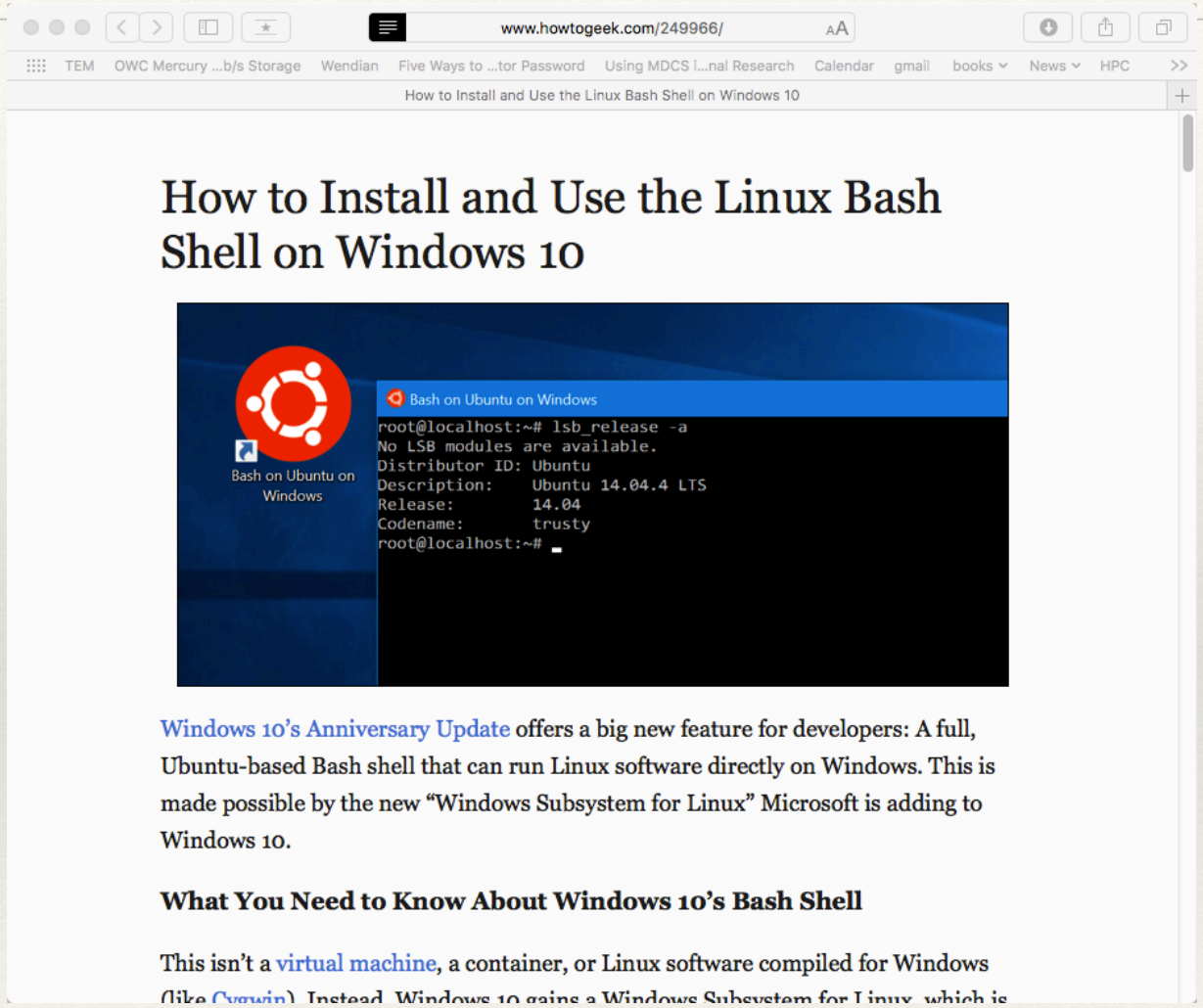
FireSSH plugin



Bash on Windows

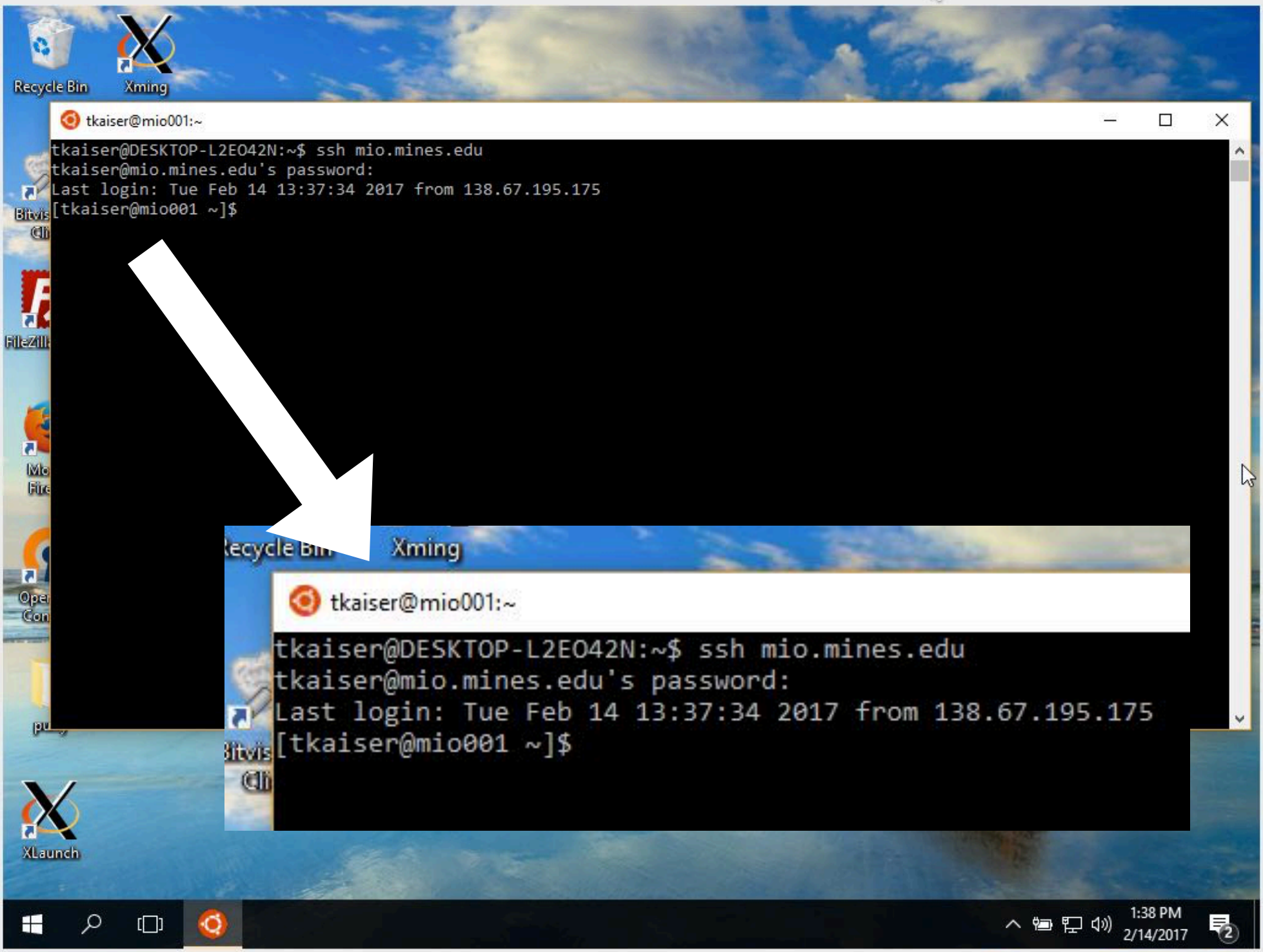
- If you are running Windows 10:
  - You can run “bash” as a process
  - Gives you a much easier life
  - Can even do ssh keys (more on this later)

Bash on Windows

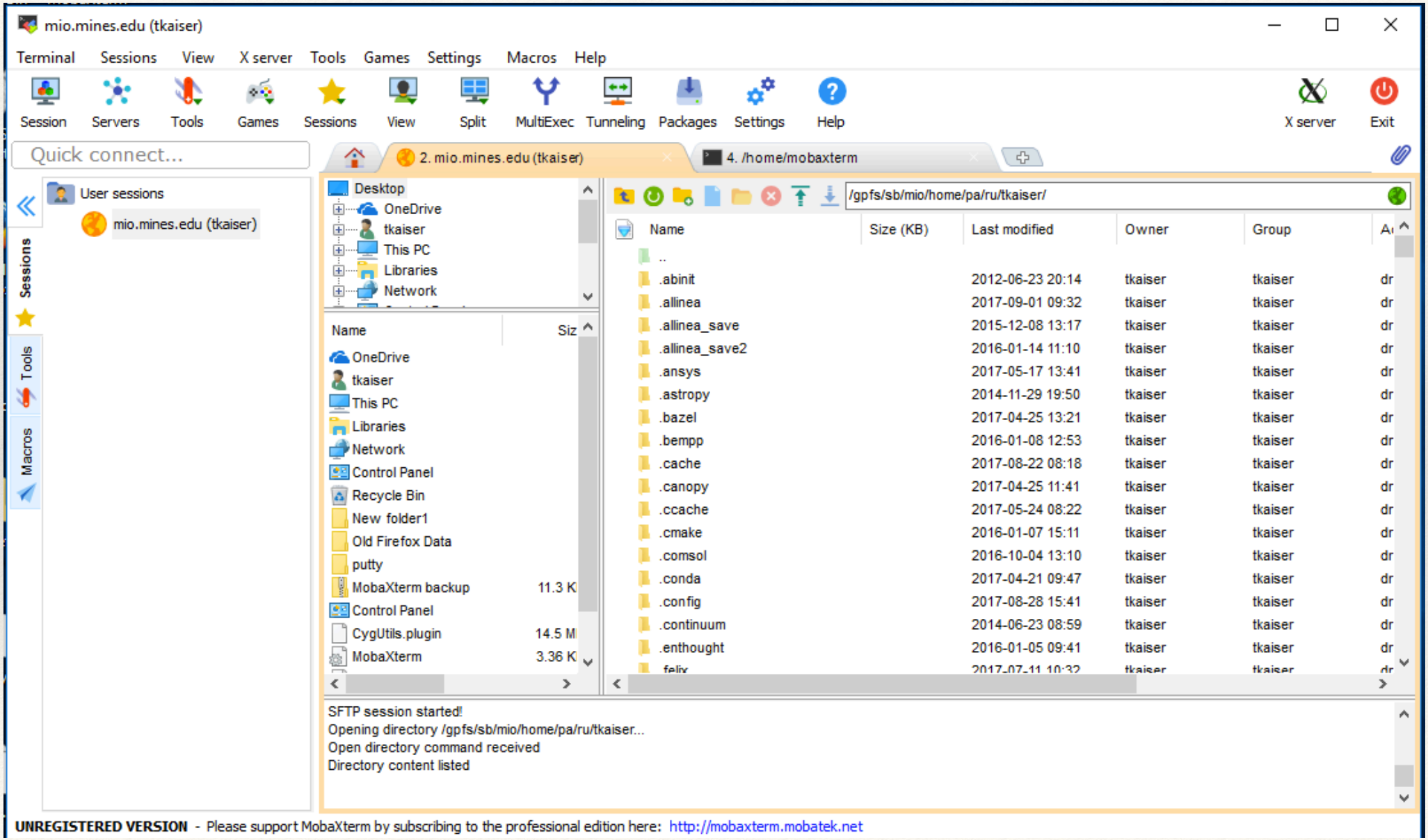


http://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/

Bash on Windows


mrbill [Running]
Recycle Bin
Xming
tkaiser@mio001:~
tkaiser@DESKTOP-L2EO42N:~$ ssh mio.mines.edu
tkaiser@mio.mines.edu's password:
Last login: Tue Feb 14 13:37:34 2017 from 138.67.195.175
[tkaiser@mio001 ~]$
Xming
tkaiser@mio001:~
tkaiser@DESKTOP-L2EO42N:~$ ssh mio.mines.edu
tkaiser@mio.mines.edu's password:
Last login: Tue Feb 14 13:37:34 2017 from 138.67.195.175
[tkaiser@mio001 ~]$
XLaunch
1:38 PM
2/14/2017
Left ⌘

MobaXterm


mio.mines.edu (tkaiser)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. mio.mines.edu (tkaiser)
4. /home/mobaxterm
User sessions
mio.mines.edu (tkaiser)
Sessions Tools Macros
Desktop
OneDrive
tkaiser
This PC
Libraries
Network
Name Siz
OneDrive
tkaiser
This PC
Libraries
Network
Control Panel
Recycle Bin
New folder1
Old Firefox Data
putty
MobaXterm backup 11.3 K
Control Panel
CygUtils.plugin 14.5 M
MobaXterm 3.36 K
/gpfs/sb/mio/home/pa/ru/tkaiser/
Name Size (KB) Last modified Owner Group
.abinit 2012-06-23 20:14 tkaiser tkaiser
.allinea 2017-09-01 09:32 tkaiser tkaiser
.allinea_save 2015-12-08 13:17 tkaiser tkaiser
.allinea_save2 2016-01-14 11:10 tkaiser tkaiser
.ansys 2017-05-17 13:41 tkaiser tkaiser
.astropy 2014-11-29 19:50 tkaiser tkaiser
.bazel 2017-04-25 13:21 tkaiser tkaiser
.bempp 2016-01-08 12:53 tkaiser tkaiser
.cache 2017-08-22 08:18 tkaiser tkaiser
.canopy 2017-04-25 11:41 tkaiser tkaiser
.ccache 2017-05-24 08:22 tkaiser tkaiser
.cmake 2016-01-07 15:11 tkaiser tkaiser
.comsol 2016-10-04 13:10 tkaiser tkaiser
.conda 2017-04-21 09:47 tkaiser tkaiser
.config 2017-08-28 15:41 tkaiser tkaiser
.continuum 2014-06-23 08:59 tkaiser tkaiser
.enthought 2016-01-05 09:41 tkaiser tkaiser
SFTP session started!
Opening directory /gpfs/sb/mio/home/pa/ru/tkaiser...
Open directory command received
Directory content listed
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

Back to ssh

- The previous few slides show programs for connecting to a Linux box using ssh
- First time: may be asked to set up ssh keys
  - Usually just hit return for the default selection
  - More on this later

ssh from a terminal window

Files Linux, more details

- The file system
  - moving around
  - listing
  - hidden files
  - “wildcards”
  - deleting
  - creating/removing directories
  - creating files
    - touch
    - edit
    - pipe
    - redirect

File System

- Tree structure for storing files:
- Most GUI interfaces to a file system show the structure as a collection of folders and subfolders
- The folders are a visual depiction of a “Directory”
- On a text based interface the structure is shown as a list of files and directories
- Directories can contain files and more directories.

Directory tree structure

- In a text based interface you are always somewhere in the tree structure
- The base of the structure is /
- “/“ is a directory that contains the whole tree
- There are commands for moving around, the structure and viewing/creating/deleting/moving/changing/listing “stuff”
- Users have a home directory which the system puts them in when they login

File system overview

Feature	Example
Moving around in the file system	cd
Listing files	ls
Where are we	pwd
Hidden files	.bashrc, .ssh
Wildcards	*,{a,b,c},[0-9]
Remove	rm, rmdir
Creating directories	mkdir

Moving around

- The primary command for moving around in the file system is “cd” - change directory
- cd . Don’t go anywhere “.” is the current directory
- cd .. Go up one level “..” is up one ../.. = two levels
- cd ~ Go to your home directory
- cd adir Go to a subdirectory “adir”
- cd adir/bdir Go to a subdirectory two levels down
- cd /u/pa/ru/tkaiser Go to an absolute location
- cd - Go back to the directory from which you came

Listing what's in a directory

- ls is the primary command for listing files in a directory
- Has many options
- ls by itself just gives names

Some ls options

```
-a, --all
       do not ignore entries starting with .

--color
       colorize the output.

-l     use a long listing format

-F, --classify
       append indicator (one of */=>@|) to entries

-r, --reverse
       reverse order while sorting

 -R, --recursive
     list subdirectories recursively

-s, --size
     print the allocated size of each file, in blocks

-S     sort by file size

-t     sort by last change date

-X     sort alphabetically by entry extension
```

ls -X

```
[tkaiser@mc2 ~]$ ls -X
after                 lib                 slurmnodes          tintel.f90
ALIAS                 local               stack               bgxlc.html
allinea               makefile            tau                 bgxlf.html
atest                 mc2                 testddt             mod.html
aun                   mc2_script          helloc.c            smap.html
auto_launch_tag       onmc2               Futil_getarg.f      util_getarg.o
b4                    privatemodules      get_inp_file.F      a.out
bashrc                remote              util_getarg.F       ddt.out
bin                   scratch             util_getenv.F       batch.qtf
bins                  scripts             color.f90
1009_210203vestalac1.tgz
ddt-script            serial              docol.f90           example.tgz
getnew                setbonk             dosin.f90
hist                  setbonk2            f90split.f90
HPM                   slurmjobs           junk.f90
[tkaiser@mc2 ~]$
```

ls -a

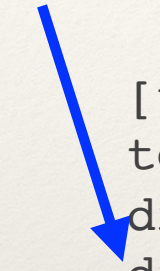
```
[tkaiser@mc2 ~]$ ls -a
.                           bins                .history            scripts
..                          .cache              HPM                 serial
1009_210203vestalac1.tgz    color.f90           junk.f90            setbonk
after                       .config             .kshrc              setbonk2
ALIAS                       .dbus               .lesshst            slurmjobs
allinea                     ddt.out             lib                 slurmnodes
.allinea                    ddt-script          local               smap.html
a.out                       docol.f90           .local              .ssh
atest                       dosin.f90           makefile            stack
aun                         .emacs              mc2                 .subversion
auto_launch_tag             example.tgz         mc2_script          tau
b4                          f90split.f90        mod.html            testddt
.bash_history               .fontconfig         .mozilla            tintel.f90
.bash_logout                Futil_getarg.f      .nedit              util_getarg.F
.bash_profile               .gconf              onmc2               util_getarg.o
bashrc                      .gconfd             .pki                util_getenv.F
.bashrc                     get_inp_file.F      privatemodules      .vim
batch.qtf                   getnew              .qt                 .viminfo
bgxlc.html                  .gnome2             .recently-used.xbel .Xauthority
bgxlf.html                  helloc.c            remote
bin                         hist                scratch
[tkaiser@mc2 ~]$
```

ls -R

```
[tkaiser@mc2 ~]$ ls -R scripts
scripts:
do1d  example  mc2_script  serial  set1

scripts/example:
aun_script  docol.f90    helloc.c  mc2_old_script
color.f90   example.tgz  makefile  mc2_script

scripts/serial:
728  a.out       fort_000001  fort_006792  mc2_script   slurm-728.out  small.py
729  do_thread   fort_006762  hello.f90    simple_test  slurm-729.out

scripts/serial/728:
bonk.out  env.728  script.728  submit

scripts/serial/729:
bonk.out  env.729  script.729  submit
[tkaiser@mc2 ~]$
```

ls -l

"d" indicates this is a directory



```
[tkaiser@mc2 ~]$ ls -l scripts/serial/
total 11152
drwxrwxr-x 2 tkaiser tkaiser     512 Jan  8 14:17 728
drwxrwxr-x 2 tkaiser tkaiser     512 Jan  8 14:18 729
-rwxrwxr-x 1 tkaiser tkaiser 6453905 Dec 23 10:44 a.out
-rw-rw-r-- 1 tkaiser tkaiser    2216 Jan  8 14:18 do_thread
-rw-rw-r-- 1 tkaiser tkaiser      43 Dec 23 10:44 fort_000001
-rw-rw-r-- 1 tkaiser tkaiser      29 Dec 23 10:44 fort_006762
-rw-rw-r-- 1 tkaiser tkaiser      43 Dec 23 10:44 fort_006792
-rw-rw-r-- 1 tkaiser tkaiser     350 Dec 23 10:44 hello.f90
-rw-rw-r-- 1 tkaiser tkaiser    2492 Dec 23 10:44 mc2_script
-rwxrwxr-x 1 tkaiser tkaiser 4926419 Dec 23 10:44 simple_test
-rw-rw-r-- 1 tkaiser tkaiser    2184 Jan  8 14:17 slurm-728.out
-rw-rw-r-- 1 tkaiser tkaiser    2194 Jan  8 14:18 slurm-729.out
-rwx------ 1 tkaiser tkaiser     351 Dec 23 10:44 small.py
[tkaiser@mc2 ~]$
```

Link

```
[tkaiser@mc2 561]$ ls -l
total 192
-rw-rw-r-- 1 tkaiser tkaiser 4670 Dec 23 14:13 env.561
-rw-rw-r-- 1 tkaiser tkaiser 2470 Dec 23 14:13 script.561
-rw-rw-r-- 1 tkaiser tkaiser 4303 Dec 23 14:13 srun_1
-rw-rw-r-- 1 tkaiser tkaiser 1818 Dec 23 14:14 srun_4
-rw-rw-r-- 1 tkaiser tkaiser 2135 Dec 23 14:14 srun_8
lrwxrwxrwx 1 tkaiser tkaiser   13 Dec 23 14:13 submit -> /bins/tkaiser
-rw-rw-r-- 1 tkaiser tkaiser  657 Dec 23 14:13 tests
[tkaiser@mc2 561]$
```

* is a wildcard for all file operations

```
[tkaiser@mc2 561]$ ls
env.561  script.561  srun_1  srun_4  srun_8  submit  tests
[tkaiser@mc2 561]$ ls srun*
srun_1  srun_4  srun_8
[tkaiser@mc2 561]$
```

```
[tkaiser@mc2 561]$ ls *561
env.561  script.561
[tkaiser@mc2 561]$
```

Other wildcard options

Show a list of files in a given directory, very useful for coping files

```
[tkaiser@mio001 ~]$ ls /u/pa/ru/tkaiser/bin/{allcast,nlist,plplot}
/u/pa/ru/tkaiser/bin/allcast  /u/pa/ru/tkaiser/bin/nlist  /u/pa/ru/tkaiser/bin/plplot
[tkaiser@mio001 ~]$
```

Show files that start with the letters between l and n or z

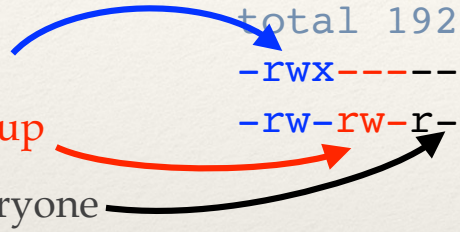
```
[tkaiser@mio001 ~]$ ls /u/pa/ru/tkaiser/bin/[l-n,z]*
/u/pa/ru/tkaiser/bin/lcpath      /u/pa/ru/tkaiser/bin/mpi_con.mod  /u/pa/ru/tkaiser/bin/noatter
/u/pa/ru/tkaiser/bin/lshtml      /u/pa/ru/tkaiser/bin/mpifnoext.h  /u/pa/ru/tkaiser/bin/node_env
/u/pa/ru/tkaiser/bin/lstohtml    /u/pa/ru/tkaiser/bin/mpihead      /u/pa/ru/tkaiser/bin/notup
/u/pa/ru/tkaiser/bin/mantohtml   /u/pa/ru/tkaiser/bin/mpi_siz.mod  /u/pa/ru/tkaiser/bin/zombie.py
/u/pa/ru/tkaiser/bin/match       /u/pa/ru/tkaiser/bin/mytop        /u/pa/ru/tkaiser/bin/zombies_sep_12
/u/pa/ru/tkaiser/bin/match.new     /u/pa/ru/tkaiser/bin/newdir     /u/pa/ru/tkaiser/bin/zombies_sep_12b
[tkaiser@mio001 ~]$
```

Show files that start with z and have two numbers in the name

```
[tkaiser@mio001 ~]$ ls /u/pa/ru/tkaiser/bin/z*[1-9][1-9]*
/u/pa/ru/tkaiser/bin/zombies_sep_12  /u/pa/ru/tkaiser/bin/zombies_sep_12b
[tkaiser@mio001 ~]$
```

Set the accessibility for a file

- There are 3 flags
  - You
  - Group
  - Everyone



```
[tkaiser@mc2 561]$ ls -l
total 192
-rwx------ 1 tkaiser tkaiser 4670 Dec 23 14:13 prg.exe
-rw-rw-r-- 1 tkaiser tkaiser 2470 Dec 23 14:13 script.561
```

- Flags have both a 3 character and numeric representation:
  - 1 - an executable program or script (binary 001)
  - 2 - writable (binary 010)
  - 4 - readable (binary 100)
- These can be added
  - 1+2+4=7= an executable program or script that is readable and writeable (binary 111)
  - 2+4= 6= a file that is readable and writeable (binary 110)

value	value	permissions
000	0	- - -
001	1	- - x
010	2	- w -
011	3	- w x
100	4	r - -
101	5	r - x
110	6	r w -
111	7	r w x

Setting accessibility - chmod

- It is possible to change accessibility based on characters but I use the numeric representation
- chmod 700 afile
  - afile is an excitable program that only you can run, read, or change
- chmod 755 afile
  - afile is an executable program that anyone can run but only change 5=4 (run) +1 (read)
  - Strange but for someone to see a directory it must have settings of at least 5 or 7
  - chmod 640 afile
    - You - read/change
    - Your group read
    - Everyone else - nothing

value	value	permissions
000	0	- - -
001	1	- - x
010	2	- w -
011	3	- w x
100	4	r - -
101	5	r - x
110	6	r w -
111	7	r w x

Online manual pages

man chmod

```
CHMOD(1)                    BSD General Commands Manual                    CHMOD(1)

NAME
     chmod -- change file modes or Access Control Lists

SYNOPSIS
     chmod [-fv] [-R [-H | -L | -P]] mode file ...
     chmod [-fv] [-R [-H | -L | -P]] [-a | +a | =a] ACE file ...
     chmod [-fhv] [-R [-H | -L | -P]] [-E] file ...
     chmod [-fhv] [-R [-H | -L | -P]] [-C] file ...
     chmod [-fhv] [-R [-H | -L | -P]] [-N] file ...

DESCRIPTION
     The chmod utility modifies the file mode bits of the listed files as specified by the mode operand. It
     may also be used to modify the Access Control Lists (ACLs) associated with the listed files.

     The generic options are as follows:

     -f      Do not display a diagnostic message if chmod could not modify the mode for file.

     -H      If the -R option is specified, symbolic links on the command line are followed.  (Symbolic
             links encountered in the tree traversal are not followed by default.)

     -h      If the file is a symbolic link, change the mode of the link itself rather than the file that
             the link points to.
:
```

man -k can be used to search for commands

Creating files

- touch afile
  - Creates and empty file called “afile”
- Piping
- The “>” symbol “redirects” or sends the output of a command into a file
- “>>” appends output
  - date > alisting
  - ls -lt /bin >> alisting
- cp - makes a copy of a file
- mv - rename or move a file

Seeing files

- The **file** command tells what type a file you have

```
[tkaiser@bluem ~]$ file alisting
alisting: ASCII text
[tkaiser@bluem ~]$
```

If a file is a text file you can do the following

- cat - types the whole file
- tail - end of a file
  - tail -f will “cat” a file as it grows
- head - beginning of a file
- less - page through a file (q to end)
- more - similar to less (q to end)
- sort - sorts a file

Some cool things

- Linux has many small tools that can be combined to do complex tasks
- You can write simple programs called “scripts” to automate common tasks
- Built in help for most commands

More Piping < , |

- The | between two Linux commands means to take the output from the first command and use it as input to the second command
  - cat alisting | sort
  - cat file | sort -u | wc
- The < between a command and a file means to use the file as input for a command
  - sort < listing

More Piping (output)

- command > file
  - puts normal output from “command” into a file
- command >& file
  - puts output and errors from a command to a file,
    - command >& errors
    - command >& /dev/null

http://www.tldp.org/LDP/abs/html/io-redirection.html
http://compgroups.net/comp.unix.shell/bash-changing-stdout/497180

More Piping (output)

- You can put errors in a file and output in another, just save errors or have both go to a file or to the terminal
- command 1> cmd.out 2> cmd.err
  - Send normal output to cmd.out and errors to cmd.err
- command 2> cmd.err
  - Send errors to a file, Normal output would go to the screen
- command > both 2>&1
  - Send errors and output to a file “both”
- command 2>&1
  - Send errors to the terminal along with the standard output. This would normally be used if you want to pipe “errors” into another command.
  - Usage example: module avail 2>&1 | sort

Removing Files

- The command for removing files is “rm”
- Syntax
  - rm anoldfile
    - Removes the anoldfile
  - rm *f90
    - Removes all files ending in f90
  - rm -rf adir afile
    - -r recursive remove (directories also)
    - -f don’t give and error if the file does not exist

A few cool commands

- echo
  - just write something a string or variable
- date
  - date - can format it
- sed
  - read a file and write a new one with changes
- nslookup
  - find an address associated with a machine name
- grep
  - find lines in a file containing a particular string

A few cool commands

- sort
  - sort files
- alias
  - make an alias for a command
- export
  - set a variable
- which
  - tells the path to a command that you might run
- wget
  - download something from a given http (web) address

File command revisited

Some file types
python script text executable
ASCII C program text
ASCII English text
ASCII English text, with CRLF line terminators
ASCII program text
ASCII text
ASCII text, with no line terminators
Bourne-Again shell script text executable
data
directory
ELF 64-bit LSB executable
gzip compressed data
HTML document text
PDF document, version 1.4
PNG image data, 1920 x 1080, 8-bit/color RGB, non-interlaced
Zip archive data, at least v1.0 to extract

The file command attempts to determine a file type from its contents. These are some of the types that can be shown.

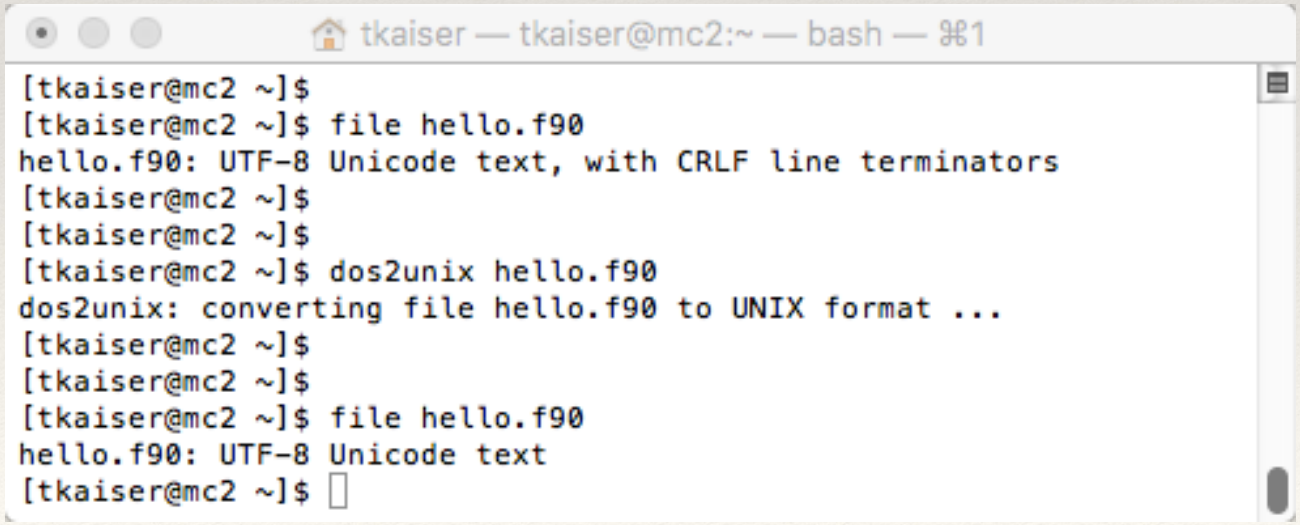
Files that have "CRLF line terminators" might not be read properly by some programs. The command dos2unix can convert these files to a readable form.

dos2unix

In DOS/Windows text files, a line break, also known as newline, is a combination of two characters: a Carriage Return (CR) followed by a Line Feed (LF). In Unix text files a line break is a single character: the Line Feed (LF). In Mac text files, prior to Mac OS X, a line break was single Carriage Return (CR) character. Nowadays Mac OS uses Unix style (LF) line breaks.

This can cause problems with data files and programs created or edited on a Windows machine. The files may not be read properly.

The command do2unix will convert a Windows text file to a linux text file.



see: http://www.computerhope.com/unix/dos2unix.htm

Let’s Do It

- ssh -Y mio.mines.edu
- ls
- ls -a
- ls /

The Environment

- You interact with the machine via a program called the shell
- Several shell programs: csh, tcsh, zsh, bash…
- We will be using bash
- When bash starts up it reads several files to set up the environment
  - .bashrc - “sourced” when you start bash
  - .bash_profile - “sourced” when you login

The Environment

- The environment is “set up” by setting various environmental variables
- The following commands will show what is set
  - export
  - printenv
  - The difference is that “export” shows them in a form that can be reused and export can also be used to set a variable

Setting a variable

```
osage:~ tkaiser$ export BONK="abcd"
osage:~ tkaiser$ printenv BONK
abcd

osage:~ tkaiser$ echo $BONK
abcd
```

declare can also set variables

```
osage:~ tkaiser$ declare -x BONK="12345"
osage:~ tkaiser$ printenv BONK
12345
```

```
[tkaiser@mc2 ~]$ export
declare -x HISTSIZE="1000"
declare -x HOME="/u/pa/ru/tkaiser"
declare -x HOSTNAME="mc2"
declare -x INCLUDE="/bgsys/drivers/ppcfloor/comm/include:/opt/ibmcmp/xlf/bg/14.1/include:/opt/
ibmcmp/vacpp/bg/12.1/include"
declare -x LANG="en_US.UTF-8"
declare -x LD_LIBRARY_PATH="/bgsys/drivers/ppcfloor/comm/lib:/opt/ibmcmp/xlf/bg/14.1/lib64:/opt/
ibmcmp/vacpp/bg/12.1/lib64"
declare -x LIBRARY_PATH="/bgsys/drivers/ppcfloor/comm/lib:/opt/ibmcmp/xlf/bg/14.1/lib64:/opt/
ibmcmp/vacpp/bg/12.1/lib64"
declare -x LOADEDMODULES="PrgEnv/IBM/VACPP/12.1.bgq:PrgEnv/IBM/XLF/14.1.bgq:PrgEnv/IBM/
default:PrgEnv/MPI/IBM/default:Core/Devel"
declare -x LOGNAME="tkaiser"
declare -x MANPATH="/opt/ibmcmp/xlf/bg/14.1/man/en_US:/opt/ibmcmp/vacpp/bg/12.1/man/en_US:/usr/
share/man"
declare -x MODULEPATH="/usr/share/Modules/modulefiles:/etc/modulefiles:/opt/modulefiles"
declare -x MODULESHOME="/usr/share/Modules"
declare -x MPI_BIN="/bgsys/drivers/ppcfloor/comm/bin/xl"
declare -x MPI_COMPILER="mpicc"
declare -x MPI_HOME="/bgsys/drivers/ppcfloor/comm"
declare -x MPI_INCLUDE="/bgsys/drivers/ppcfloor/comm/include"
declare -x MPI_LIB="/bgsys/drivers/ppcfloor/comm/lib"
declare -x MPI_SUFFIX="_mpich"
declare -x OLDPWD
declare -x PATH="/bgsys/drivers/ppcfloor/comm/bin/xl:/opt/ibmcmp/xlf/bg/14.1/bin:/opt/ibmcmp/vacpp/
bg/12.1/bin:/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/u/
pa/ru/tkaiser/bin"
declare -x PWD="/u/pa/ru/tkaiser"
declare -x SCRATCH="/scratch/tkaiser"
declare -x SHELL="/bin/bash"
declare -x USER="tkaiser"
```

Important variables

- PATH
  - Where to look for programs to run
- LD_LIBRARY_PATH
  - Where to look for libraries to use when running programs
- MANPATH
  - Where to look for man (manual) pages

Setting up your environment

- You can run “export” form the command line
- If you want to have an environment set every time you login or start bash you set that in
  - .bashrc
  - .bash_profile

Example…

- Say I want
  - PATH to include ~/bin and “.”
  - LD_LIBRARY_PATH to include ~/lib

Example...

```
osage:~ tkaiser$ ssh petra
tkaiser@petra's password:
Last login: Wed Mar 12 08:59:58 2014 from osage.mines.edu
[tkaiser@petra ~]$ printenv PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin
[tkaiser@petra ~]$ printenv LD_LIBRARY_PATH
[tkaiser@petra ~]$
```

Original .bashrc

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
```

Becomes

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions

export PATH=.:~/bin:$PATH

export LD_LIBRARY_PATH=~/lib:$LD_LIBRARY_PATH
```

On next login…

```
osage:~ tkaiser$ ssh petra
tkaiser@petra's password:
Last login: Wed Mar 12 08:59:58 2014 from osage.mines.edu
osage:~ tkaiser$
osage:~ tkaiser$

[tkaiser@petra ~]$ printenv PATH
.:/home/tkaiser/bin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin
osage:~ tkaiser$
osage:~ tkaiser$
[tkaiser@petra ~]$ printenv LD_LIBRARY_PATH
/home/tkaiser/lib
[tkaiser@petra ~]$
```

Modules

- Some systems (bluem, Mio001, AuN, Mc2) have sets of variables combined into modules
- To see what modules are available run
  - module avail
- To load the set you run the load module command:
  - module load
- You can put module load commands in .bashrc

Modules

```
[tkaiser@mio001 ~]$ module avail

-------- /usr/share/Modules/modulefiles --------
dot         module-cvs  module-info modules     null        use.own     utility

-------- /opt/modulefiles --------
PrgEnv/intel/13.0.1                  impi/gcc/4.1.1
PrgEnv/intel/default                 impi/intel/4.1.1
PrgEnv/libs/fftw/gcc/3.3.3           openmpi/gcc/1.6.5
PrgEnv/libs/fftw/intel/3.3.3         openmpi/gcc/default
PrgEnv/libs/opencl/1.2               openmpi/intel/1.6.5
PrgEnv/python/Enthought/2.7.2_v7.1-2 openmpi/intel/1.6.5_test
ansys/fluent/15.0                    openmpi/intel/default
[tkaiser@mio001 ~]$
[tkaiser@mio001 ~]$

[tkaiser@mio001 ~]$ which mpicc
/opt/openmpi/1.6.5/intel/bin/mpicc
[tkaiser@mio001 ~]$
[tkaiser@mio001 ~]$

[tkaiser@mio001 ~]$ module load impi/intel/4.1.1
[tkaiser@mio001 ~]$ which mpicc
/opt/intel/impi/4.1.1.036/intel64/bin/mpicc
```

Moving files to/from machines

- scp - secure copy
- related to ssh
- Full syntax
  - scp source destination

```
scp username@machine:path_to_file username@machine:path_to_file
```

  - Can usually shorten this
  - I have a helpful utility /opt/utility/scpath

```
[tkaiser@aun002 ~]$ /opt/utility/scpath
tkaiser@aun002.mines.edu:/u/pa/ru/tkaiser

[tkaiser@aun002 ~]$ /opt/utility/scpath binary.f90
tkaiser@aun002.mines.edu:/u/pa/ru/tkaiser/binary.f90
[tkaiser@aun002 ~]$
```

Shorter forms

- scp afile tkaiser@bluem:/u/pa/ru/tkaiser/tmp
  - copy a local file to bluem
- scp afile bluem:/u/pa/ru/tkaiser/tmp
  - same as above
- scp afile bluem:~
  - copy to home directory
- scp bluem:/u/pa/ru/tkaiser/tmp/afile .
  - copy from bluem to local directory
- scp -r bluem:/u/pa/ru/tkaiser/tmp .
  - copy a complete directory to your local directory

A useful utility

- /opt/utility/scpath
- Gives full paths for scp
- scp then becomes a copy/past activity

```
[tkaiser@bluem bins]$ ls
abinit          amber       examples.tgz  gromacs  matrix  nwchem  quick    wu
abinit-6.10.2   enthought   fft           grow     memory  petsc   siesta
acc.tgz         examples    fftw          guide    nbody   ppong   utility
[tkaiser@bluem bins]$
[tkaiser@bluem bins]$
[tkaiser@bluem bins]$ scpath
tkaiser@bluem:/u/pa/ru/tkaiser/remote/aun/bins
[tkaiser@bluem bins]$
[tkaiser@bluem bins]$
[tkaiser@bluem bins]$ scpath amber
tkaiser@bluem:/u/pa/ru/tkaiser/remote/aun/bins/amber
[tkaiser@bluem bins]$
[tkaiser@bluem bins]$ scpath *tgz
tkaiser@bluem:/u/pa/ru/tkaiser/remote/aun/bins/acc.tgz
tkaiser@bluem:/u/pa/ru/tkaiser/remote/aun/bins/examples.tgz
[tkaiser@bluem bins]$
```

scp GUI clients

- These are very useful in the context of editing a file
- There are a number of good ones:
  - WinSCP (Windows)
  - Bitvice (Windows)
  - Putty (cross platform, mostly Windows)
  - MobaXterm (Windows)
  - Filezilla (cross platform)
  - FireFTP (Firefox extension) See: FireSSH also
  - Yummy (OSX)
  - Fetch (OSX)

Editing

- nano
- gedit (GUI)
- gvim (GUI)
- emacs (GUI)
- Remote editing
- Not covered
  - vi (Available on every Linux box)
  - emacs (Text based version of emacs)

Nano

- Text based editor
- Relatively easy to use
- Online help
  - Copy at http://hpc.mines.edu/nano.html
  - ^ - implies the control key

Nano Important commands

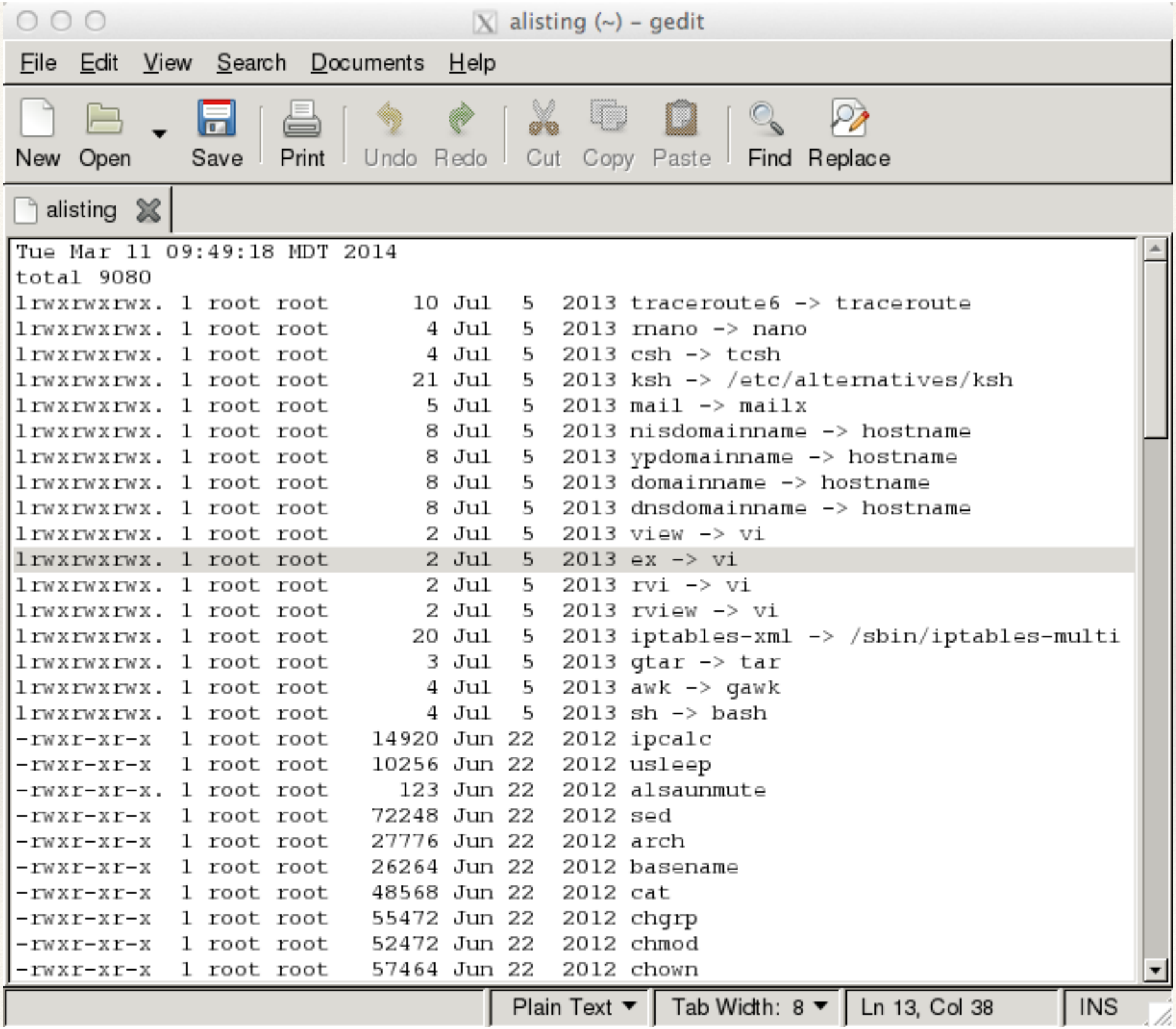
- ^w - find
- ^\ - find and replace
- ^k - cut text (aline or marked text)
- ^u - paste text
- ^O - save the file
- ^X - quit
- ^^ - mark text for cutting. In this case the second ^ is not the control character but the “real” ^, usually shift 6
- ^G - show the online help

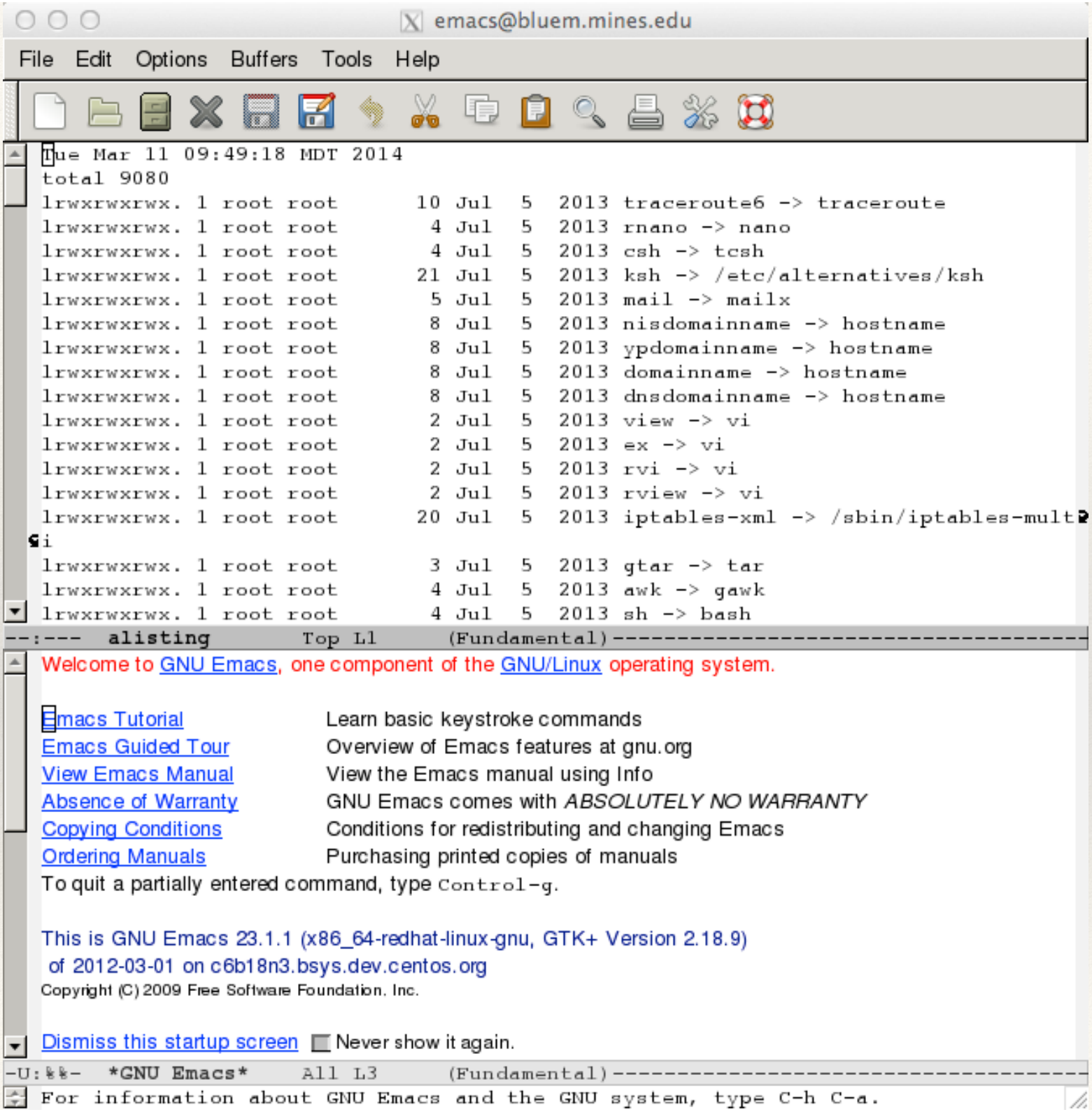
Nano Screen Dump

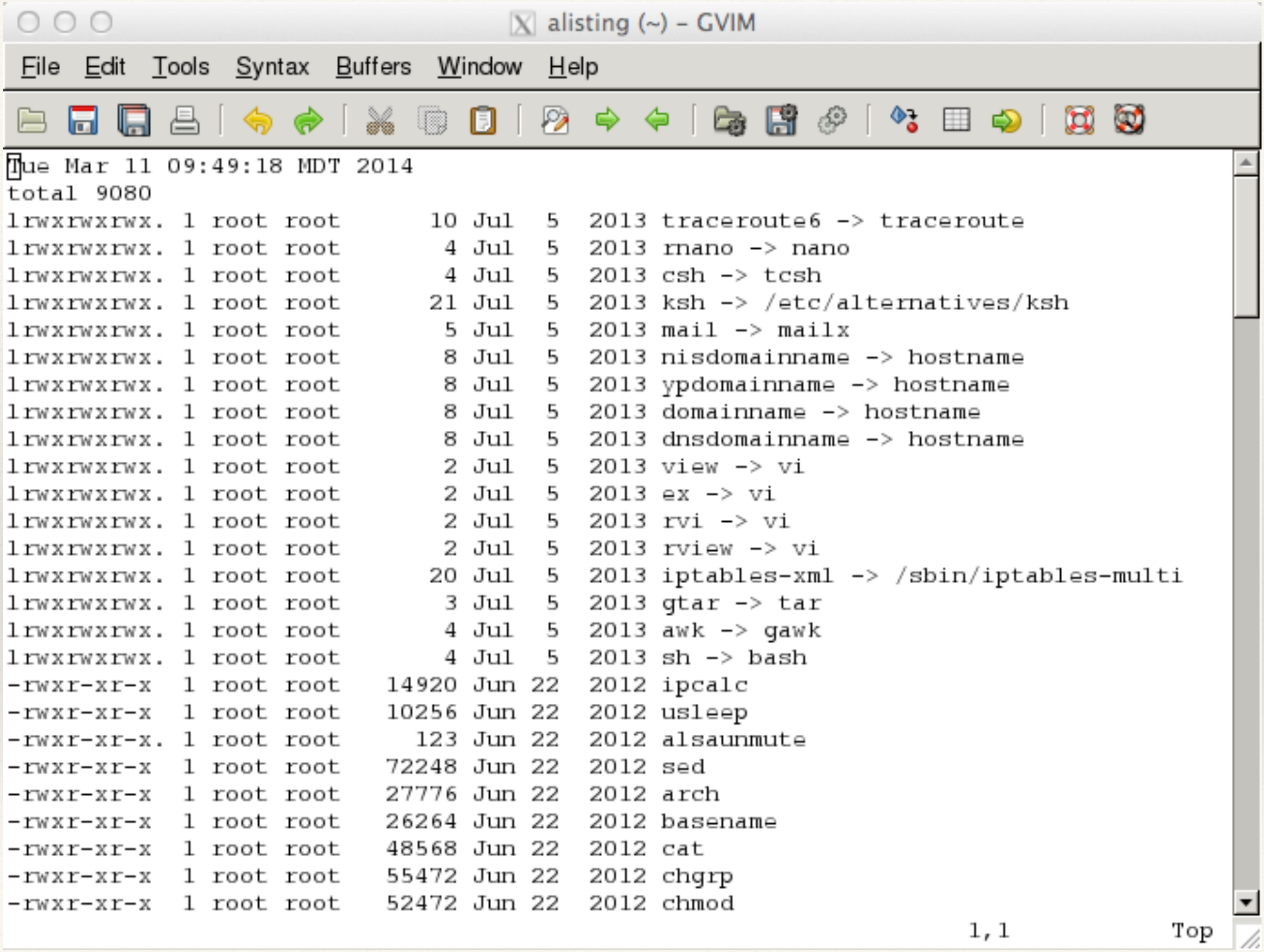
GUI base editors

- These are available on Mio,BlueM, AuN, Mc2
  - gedit
  - gvim
  - emacs
  - Require X-windows but gvim and emacs will fall back to a text version
  - Launch them in the background with the & option and put errors into /dev/null

```
[tkaiser@bluem ~]$ gedit alisting   >& /dev/null &
```


alisting (~) - gedit
File Edit View Search Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
alisting
Tue Mar 11 09:49:18 MDT 2014
total 9080
lrwxrwxrwx. 1 root root 10 Jul 5 2013 traceroute6 -> traceroute
lrwxrwxrwx. 1 root root 4 Jul 5 2013 rnano -> nano
lrwxrwxrwx. 1 root root 4 Jul 5 2013 csh -> tcsh
lrwxrwxrwx. 1 root root 21 Jul 5 2013 ksh -> /etc/alternatives/ksh
lrwxrwxrwx. 1 root root 5 Jul 5 2013 mail -> mailx
lrwxrwxrwx. 1 root root 8 Jul 5 2013 nisdomainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 ypdomainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 domainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 dnsdomainname -> hostname
lrwxrwxrwx. 1 root root 2 Jul 5 2013 view -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 ex -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 rvi -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 rview -> vi
lrwxrwxrwx. 1 root root 20 Jul 5 2013 iptables-xml -> /sbin/iptables-multi
lrwxrwxrwx. 1 root root 3 Jul 5 2013 gtar -> tar
lrwxrwxrwx. 1 root root 4 Jul 5 2013 awk -> gawk
lrwxrwxrwx. 1 root root 4 Jul 5 2013 sh -> bash
-rwxr-xr-x 1 root root 14920 Jun 22 2012 ipcalc
-rwxr-xr-x 1 root root 10256 Jun 22 2012 usleep
-rwxr-xr-x. 1 root root 123 Jun 22 2012 alsaunmute
-rwxr-xr-x 1 root root 72248 Jun 22 2012 sed
-rwxr-xr-x 1 root root 27776 Jun 22 2012 arch
-rwxr-xr-x 1 root root 26264 Jun 22 2012 basename
-rwxr-xr-x 1 root root 48568 Jun 22 2012 cat
-rwxr-xr-x 1 root root 55472 Jun 22 2012 chgrp
-rwxr-xr-x 1 root root 52472 Jun 22 2012 chmod
-rwxr-xr-x 1 root root 57464 Jun 22 2012 chown
Plain Text
Tab Width: 8
Ln 13, Col 38
INS


emacs@bluem.mines.edu
File Edit Options Buffers Tools Help
Tue Mar 11 09:49:18 MDT 2014
total 9080
lrwxrwxrwx. 1 root root 10 Jul 5 2013 traceroute6 -> traceroute
lrwxrwxrwx. 1 root root 4 Jul 5 2013 rnano -> nano
lrwxrwxrwx. 1 root root 4 Jul 5 2013 csh -> tcsh
lrwxrwxrwx. 1 root root 21 Jul 5 2013 ksh -> /etc/alternatives/ksh
lrwxrwxrwx. 1 root root 5 Jul 5 2013 mail -> mailx
lrwxrwxrwx. 1 root root 8 Jul 5 2013 nisdomainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 ypdomainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 domainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 dnsdomainname -> hostname
lrwxrwxrwx. 1 root root 2 Jul 5 2013 view -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 ex -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 rvi -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 rview -> vi
lrwxrwxrwx. 1 root root 20 Jul 5 2013 iptables-xml -> /sbin/iptables-multi
lrwxrwxrwx. 1 root root 3 Jul 5 2013 gtar -> tar
lrwxrwxrwx. 1 root root 4 Jul 5 2013 awk -> gawk
lrwxrwxrwx. 1 root root 4 Jul 5 2013 sh -> bash
--:--- alisting Top L1 (Fundamental)
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
Emacs Tutorial Learn basic keystroke commands
Emacs Guided Tour Overview of Emacs features at gnu.org
View Emacs Manual View the Emacs manual using Info
Absence of Warranty GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions Conditions for redistributing and changing Emacs
Ordering Manuals Purchasing printed copies of manuals
To quit a partially entered command, type Control-g.
This is GNU Emacs 23.1.1 (x86_64-redhat-linux-gnu, GTK+ Version 2.18.9)
of 2012-03-01 on c6b18n3.bsys.dev.centos.org
Copyright (C) 2009 Free Software Foundation, Inc.
Dismiss this startup screen
Never show it again.
-U:%%- *GNU Emacs* All L3 (Fundamental)
For information about GNU Emacs and the GNU system, type C-h C-a.


alisting (~) - GVIM
File Edit Tools Syntax Buffers Window Help
Tue Mar 11 09:49:18 MDT 2014
total 9080
lrwxrwxrwx. 1 root root 10 Jul 5 2013 traceroute6 -> traceroute
lrwxrwxrwx. 1 root root 4 Jul 5 2013 rnano -> nano
lrwxrwxrwx. 1 root root 4 Jul 5 2013 csh -> tcsh
lrwxrwxrwx. 1 root root 21 Jul 5 2013 ksh -> /etc/alternatives/ksh
lrwxrwxrwx. 1 root root 5 Jul 5 2013 mail -> mailx
lrwxrwxrwx. 1 root root 8 Jul 5 2013 nisdomainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 ypdomainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 domainname -> hostname
lrwxrwxrwx. 1 root root 8 Jul 5 2013 dnsdomainname -> hostname
lrwxrwxrwx. 1 root root 2 Jul 5 2013 view -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 ex -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 rvi -> vi
lrwxrwxrwx. 1 root root 2 Jul 5 2013 rview -> vi
lrwxrwxrwx. 1 root root 20 Jul 5 2013 iptables-xml -> /sbin/iptables-multi
lrwxrwxrwx. 1 root root 3 Jul 5 2013 gtar -> tar
lrwxrwxrwx. 1 root root 4 Jul 5 2013 awk -> gawk
lrwxrwxrwx. 1 root root 4 Jul 5 2013 sh -> bash
-rwxr-xr-x 1 root root 14920 Jun 22 2012 ipcalc
-rwxr-xr-x 1 root root 10256 Jun 22 2012 usleep
-rwxr-xr-x. 1 root root 123 Jun 22 2012 alsaunmute
-rwxr-xr-x 1 root root 72248 Jun 22 2012 sed
-rwxr-xr-x 1 root root 27776 Jun 22 2012 arch
-rwxr-xr-x 1 root root 26264 Jun 22 2012 basename
-rwxr-xr-x 1 root root 48568 Jun 22 2012 cat
-rwxr-xr-x 1 root root 55472 Jun 22 2012 chgrp
-rwxr-xr-x 1 root root 52472 Jun 22 2012 chmod
1,1 Top

Remote Editing

- Idea: Copy the file to your desktop machine and edit it locally. Then send it back
- Some (most) of the scp GUI clients support this almost automatically
- Examples:
  - Filezilla
  - Yummy (OSX)
  - FireFTP
- You can double click on a file to edit it.
  - May need to select your local editor
  - After that, it is automatic

Back to ssh

Local ssh pages

- Setting up ssh, including putty
  - http://geco.mines.edu/ssh/
- Tunneling
  - http://geco.mines.edu/ssh/tunneling.html
  - http://hpc.mines.edu/bluem/transfer.html#scp
  - http://hpc.mines.edu/bluem/multistage.html

ssh

- Reads a local configuration file ~/.ssh/config (if it exists)
  - Alias
  - Special password settings
  - Tunnels
- Sets up an encrypted connection between your local and remote machines
- “Normally” asks for a password (MultiPass)
- Opens up a session on the remote host in which you can enter commands
- Type exit to quit

ssh keys

- Setting up keys
- Keys are like two part passwords
  - Private part - on the machine you are coming from
  - Public part - on the machine you are going to
    - You can give someone your public key
      - They put it on a machine in:
        - ~.ssh/authorized_keys
      - You now have access

ssh keys

- Private keys have a pass phrase which must be entered to allow its use
  - Can have a pass phrase that you enter like a password every time
  - Can have a blank pass phrase which will allow getting on to a machine without needing a password. (This is a lot more common than you think.)
  - Can enter a pass phrase with a timeout feature
- Once a pass phrase is validated you can use it on all machines that have the public key

More on keys…

- The command to generate a key set is ssh-keygen
- -t option tells what "type" of key
  - ssh-keygen -tdsa
- Keys are normally stored in a hidden directory ~/.ssh
- You can give a key set a non-default name
  - You can associate a key set with a machine in the file ~/.ssh/config

More on keys…

```
osage:~ tkaiser$ ssh-keygen -tdsa
Generating public/private dsa key pair.
Enter file in which to save the key (/Users/tkaiser/.ssh/id_dsa): \
/Users/tkaiser/.ssh/arock
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/tkaiser/.ssh/arock.
Your public key has been saved in /Users/tkaiser/.ssh/arock.pub.
The key fingerprint is:
af:62:a1:01:42:03:b2:f9:78:36:24:d2:18:a3:82:70 tkaiser@osage.Mines.EDU
The key's randomart image is:
+--[ DSA 1024]----+
|B E              |
|=@               |
|@ +              |
|+=.              |
|..=.    S        |
| o .. .  .       |
|     o .  .      |
|    . o  .       |
|     . ..        |
+-----------------+
osage:~ tkaiser$
```

A slight digression - .ssh/config

```
Host petra petra.mines.edu peter
HostName 138.67.4.29
User tkaiser
Identityfile2 ~/.ssh/arock
```

When you ssh to petra, petra.mines.edu or peter you:

- Connect to a machine at 138.67.4.29
- Username is tkaiser
- Use the keys found in ~/.ssh/arock

Set up keys and copy them to "bluem"

Create a key set:

```
osage:.ssh tkaiser$ ssh-keygen -tdsa
Generating public/private dsa key pair.
Enter file in which to save the key (/Users/tkaiser/.ssh/id_dsa): /
Users/tkaiser/.ssh/brock
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/tkaiser/.ssh/brock.
Your public key has been saved in /Users/tkaiser/.ssh/brock.pub.
The key fingerprint is:
e9:bb:b4:20:2c:af:fc:5c:4d:e7:c4:50:3c:82:db:64
tkaiser@osage.mines.edu
osage:.ssh tkaiser$


osage:.ssh tkaiser$ ls -lt brock*
-rw-------  1 tkaiser  staff  751 Mar 12 11:24 brock
-rw-r--r--  1 tkaiser  staff  613 Mar 12 11:24 brock.pub
osage:.ssh tkaiser$
```

Set up keys and copy them to “bluem”

Copy the public key to bluem

```
cd ~/.ssh
cat brock.pub | ssh bluem.mines.edu "cat >> .ssh/authorized_keys"
tkaiser@138.67.132.239's password:
osage:.ssh tkaiser$
```

Tell ssh to use our new key

Get the address for bluem

```
osage:.ssh tkaiser$ nslookup bluem
Server:         138.67.1.2
Address:  138.67.1.2#53

Name:  bluem.mines.edu
Address: 138.67.132.239
```

Create our ~/.ssh/config with the following lines

```
Host bluem bluem.mines.edu
HostName 138.67.132.239
User tkaiser
Identityfile2 ~/.ssh/brock
```

Next time you login…

- You will be asked for a pass phrase instead of a pass word
- What has this bought you?
  - You can validate a key for some time and you will not need to reenter it until the time expires
  - This validates a key for 8 hours:

```
ssh-add -t 28800 ~/.ssh/brock
```

Here it is…

```
osage:.ssh tkaiser$ ssh-add -t 28800 ~/.ssh/brock
Enter passphrase for /Users/tkaiser/.ssh/brock:
Identity added: /Users/tkaiser/.ssh/brock (/Users/tkaiser/.ssh/brock)
Lifetime set to 28800 seconds
```

```
osage:.ssh tkaiser$ ssh bluem
Last login: Wed Mar 12 11:30:23 2014 from osage.mines.edu
[tkaiser@bluem ~]$
```

You will want to add the following to your .bashrc file

```
alias keys="ssh-add -t 28800 ~/.ssh/brock"
alias killkeys="ssh-add -D"
```

A common problem with ssh

- ssh is very particular about the permissions setting on its files
- Private key files must be only readable by the user
- The . ssh directory must be only readable by the user
- Public stuff can but need not be readable by all
- The setting below work

```
osage:~ tkaiser$ ls -dal .ssh
drwx------  14 tkaiser  staff  476 Jun  4 08:54 .ssh

osage:~ tkaiser$ cd .ssh
osage:.ssh tkaiser$ ls -l
total 112
-rw-------  1 tkaiser  staff    751 Jun  4 08:54 arock
-rw-r--r--  1 tkaiser  staff    613 Jun  4 08:54 arock.pub
-rw-r--r--  1 tkaiser  staff   1222 Apr  2  2014 authorized_keys
-rw-------  1 tkaiser  staff   1358 Jan 21 12:28 config
osage:.ssh tkaiser$
```

More ~/.ssh/config magic

- Motivation
  - Make your life easier
- Tunneling:
  - Get to one machine by going through another
  - Second machine might only be accessible via the first
    - Mc2 and AuN can only be seen from bluem
  - Machine might be behind a firewall
    - Mio, BlueM
    - Most campus machines

A simple tunnel

- ssh to golden goes to bluem and then is forwarded to aun.mines.edu
- ssh to energy goes to bluem and then is forwarded to mc2.mines.edu

```
Host golden
ProxyCommand ssh bluem.mines.edu nc 2>/dev/null aun.mines.edu %p

Host energy
ProxyCommand ssh bluem.mines.edu nc 2>/dev/null mc2.mines.edu %p
```

Getting to bluem from off campus

This would go on your machine at home:

- ssh to bluem from off campus goes to imagine.mines.edu and then is forwarded to bluem.mines.edu

```
Host bluem
ProxyCommand ssh imagine.mines.edu nc 2>/dev/null bluem.mines.edu %p
```

We can combine tunnels

This would go on your machine at home to get to
AuN or Mc2

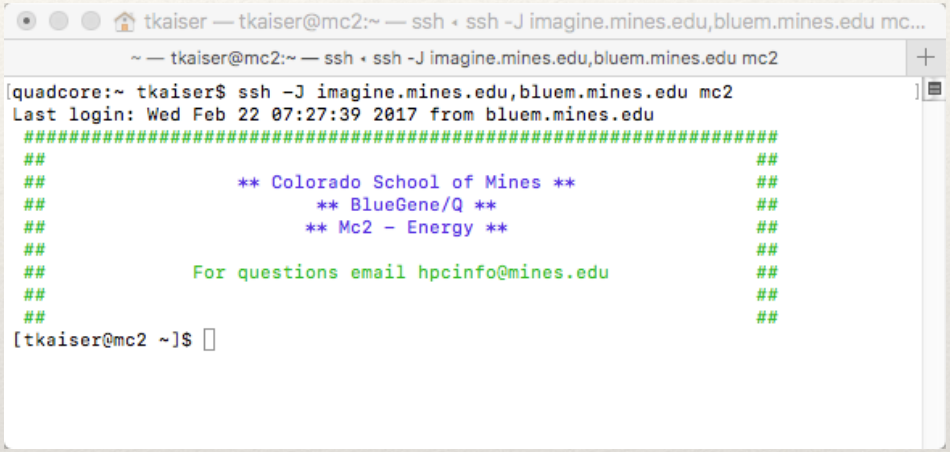
- ssh to energy from off campus goes to imagine.mines.edu and then is forwarded to bluem.mines.edu then finally to mc2.mines.edu

```
Host energy
ProxyCommand ssh step2 nc 2>/dev/null mc2.mines.edu %p

Host golden
ProxyCommand ssh step2 nc 2>/dev/null aun.mines.edu %p

Host step2
ProxyCommand ssh step1 nc 2>/dev/null bluem.mines.edu %p

Host step1
Hostname imagine.mines.edu
```

```
ssh -J imagine,bluem mc2
```

Single Line Tunnels

Newer versions of ssh support a simplified tunneling method from the command line. You can use the -J option to tunnel and specify multiple hops separated by commas.



Example: To connect to mc2 from off campus, tunneling through imagine and bluem you can:

```
ssh -J imagine.mines.edu,bluem mc2
```

From the manpage:

```
-J [user@]host[:port]
        Connect to the target host by first making a ssh connection to
        the jump host and then establishing a TCP forwarding to the ulti-
        mate destination from there.  Multiple jump hops may be specified
        separated by comma characters.  This is a shortcut to specify a
        ProxyJump configuration directive.
```

Automatically forward X11 connections

```
ForwardAgent yes
ForwardX11 yes
```

```
PubkeyAcceptedKeyTypes=+ssh-dss
```

For new version of MacOS

Apple updated the ssh version in Mac OS 10.12 and the installed version does not automatically use all of the keys in your .ssh/config file. You can force it to recognize your keys by adding the desired key types to your .ssh/config file. Add the following line to your config file.

```
PubkeyAcceptedKeyTypes=+ssh-dss
```

An obscure feature

```
Host bluem bluem.mines.edu
HostName 138.67.132.239
User tkaiser
Identityfile2 ~/.ssh/brock
ControlMaster auto
ControlPath   /Users/tkaiser/.ssh/tmp/%h_%p_%r
```

After you have one login session on a machine any new connections will get piped transparently through the first connection.

If you have two part authentication this might save you some work

A few local commands in /opt/utilities

- greenbar
  - Makes *html files from tab delimited lists
- tymer
  - A timing script
- scpath
  - Returns a fully qualified path for scp
- tarup, backup, zipup
  - creates date stamped tar, tgz, zip files
- jlines
  - combines lines from output
- xtest
  - does X-Windows work

Building Programs

- Compilers
- make
- configure
- cmake

Compilers - build programs from source

- Primary language of High Performance Computing
  - Fortran (90,2000,2003,77)
  - C
  - C++
- There are special versions of these for parallel applications
- Need to match the machine

X86 compilers (Mio/Aun)

- Intel
  - ifort
  - icc
  - icpc
- gnu
  - gfortran
  - gcc
  - g++
- Portland Group
  - pgf77,pgf90, pgf95
  - pgc
  - pgc++
- NAG
  - nagfor

Power Compilers (Mc2)

- gnu
  - gfortran
  - gcc
  - g++

IBM Fortran Compilers:

- bgxlf2003_r
- bgxlf2008
- bgxlf2008_r
- bgxlf90_r
- bgxlf95_r
- bgxlf_r
- bgxlf2003
- bgxlf90
- bgxlf95
- bgxlf

IBM "C" Compilers:

- bgxlc++
- bgxlc_r
- bgxlc++_r
- bgxlC_r
- bgxlC
- bgxlc

http://hpc.mines.edu/bgq/compilers/

Note: The compute nodes on Mc2 have different processors than the head node so programs compiled for one might not work on the other

Compiling

Good idea to build/test with multiple versions of compilers
Start with optimization level -O0
Normal good optimization level is -O3

```
[tkaiser@aun001 ~]$ ifort -O0 stringit.f90 -o stringit
[tkaiser@aun001 ~]$ ls -lt stringit*
-rwxrwxr-x 1 tkaiser tkaiser 668896 Mar 12 12:37 stringit
-rw-rw-r-- 1 tkaiser tkaiser    551 Oct  3 13:42 stringit.f90
[tkaiser@aun001 ~]$ ./stringit
```

make

- Make is a system for managing the building of applications
- Reads a makefile
  - dependancies
  - instructions
- Calls compilers and similar software to do the build

```
L1= charles.o darwin.o ga_list_mod.o global.o init.o laser_new.o
L2= mods.o more_mpi.o mpi.o numz.o  unique.o wtime.o

OPT= -O3 -free
SOPT=
LINK= -lesslbg -L/bgsys/ibm_essl/prod/opt/ibmmath/lib64

PF90=mpixlf90_r

darwin: $(L1) $(L2)
    $(PF90) $(SOPT)  $(L1) $(L2) $(LINK) -o darwin

.f.o:
    $(PF90) $(SOPT) $(OPT) -c  $<

wtime.o : wtime.c
    $(CC) -DWTIME=wtime -c wtime.c

mpi.o: mpi.f

numz.o:numz.f

more_mpi.o: more_mpi.f numz.o mpi.o
```

```
charles.o: charles.f mods.o global.o more_mpi.o mpi.o numz.o

darwin.o: darwin.f ga_list_mod.o global.o more_mpi.o mpi.o numz.o mods.o

ga_list_mod.o: ga_list_mod.f

global.o: global.f

init.o: init.f global.o more_mpi.o mpi.o numz.o

laser_new.o: laser_new.f ga_list_mod.o  global.o  more_mpi.o mpi.o numz.o

mods.o: mods.f mpi.o numz.o

unique.o:unique.f mpi.o numz.o

clean:
    /bin/rm -f *o *mod $(L1b) $(L2b)
```

configure & cmake

- configure and cmake are utilities for creating makefile
- Idea:
  - A person that creates an application also creates a configure or cmake file
  - configure or cmake are run to create a make file
  - make is run to build the application
- Ideal world:
  - configure and cmake discover enough about your system to create a working makefile
  - You “may” want to specify options to tune to your system

Python

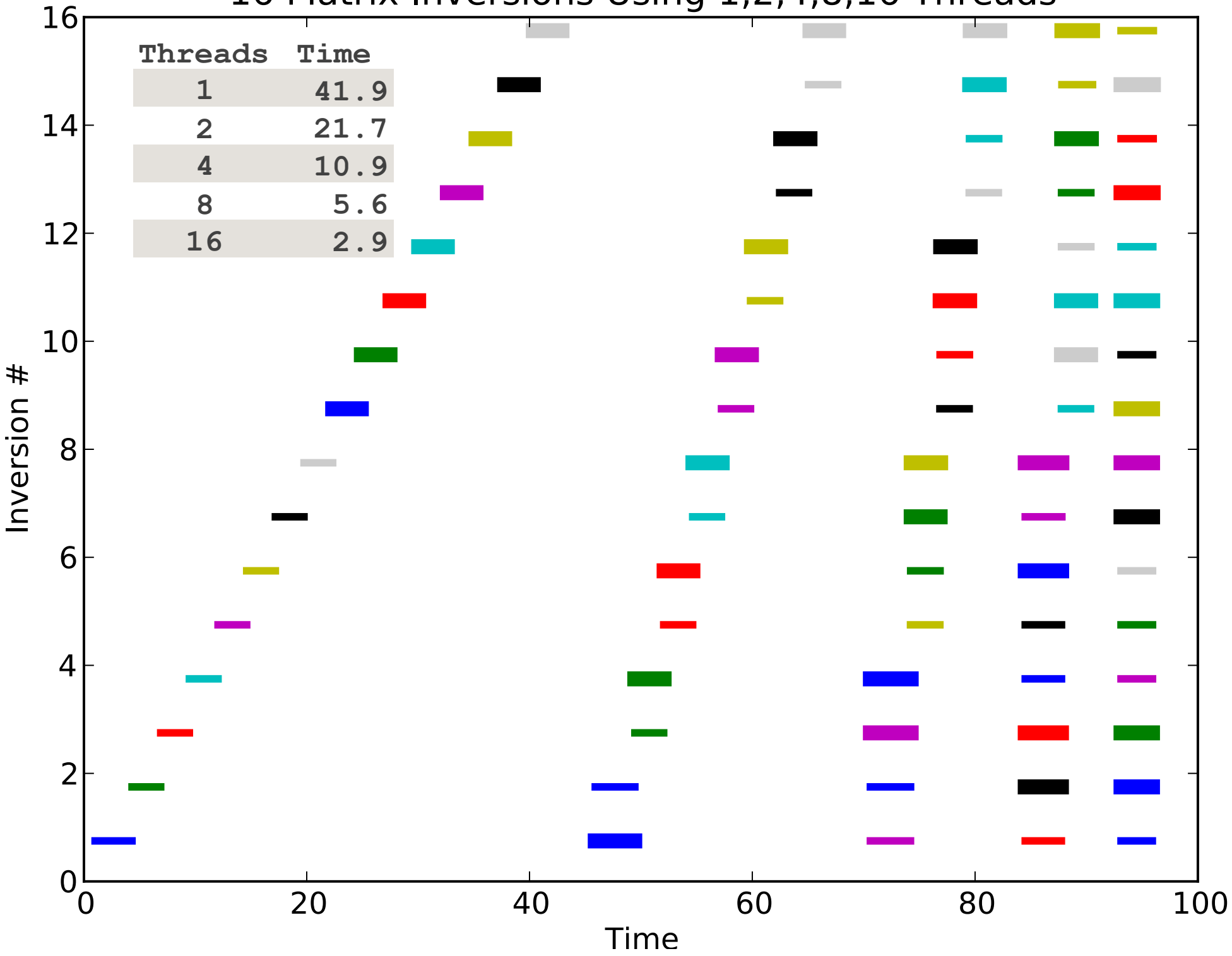
- Python is a scripting/programming language for quick tasks
- Good mixture of numeric and string (text) processing capabilities
- Easy to learn and use
- Can be run interactively
- Can be used like a calculator
- GUI and Graphics libraries
- http://www.python.org

HPC and Parallel Programming

- Concept is simple
  - If a problem takes N hours on 1 processor why not run it on N processors and finish in an hour?
  - Has all of the advantages and disadvantages of working on a committee
- Mc2 - 8192 processors in 512 nodes
- AuN - 2304 processors in 144 nodes
- Programming across multiple nodes and processors on a node requires special languages and/or compilers

Programming

- Thread programming for cores on a node
- Message passing for programming using multiple nodes
- Hybrid - using threads on node and message passing between nodes.


16 Matrix Inversions Using 1,2,4,8,16 Threads
Threads Time
1 41.9
2 21.7
4 10.9
8 5.6
16 2.9
Inversion #
Time
0
2
4
6
8
10
12
14
16
0
20
40
60
80
100

Writing Scripts

- A script is a collection of commands put in a file
- A mini program
- There are many scripting languages…
- bash, perl **python**, csh…
- Here we talk about bash

Bash

- Default shell on CSM machines
- Used to interact with the machine, run commands
- Bash commands can be run interactively or put in a script file
- A script file is really a "simple"
  - Program
  - List of commands
- First we discuss some features of bash

http://www.tldp.org/LDP/Bash-Beginners-Guide/html/
http://linuxconfig.org/bash-scripting-tutorial

Notes on Commands

- > is used to sent output to a file (date > mylisting)
- >> append output to a file (ls >> mylisting)
- >& send output and error output to a file
- The ; can be used to combine multiline commands on a single line. Thus the following are equivalent

```
date ; echo “line 2” ; date
```

```
date
echo “line 2”
date
```

Notes on Commands

- Putting commands in ` ` returns the output of a command into a variable
- Can be use create a list with other commands such as “for loops”

```
myf90=`ls *f90`
echo $myf90
doint.f90 fourd.f90 tintel.f90 tp.f90 vect.f90
```

```
np=`expr 3 + 4`
np=`expr $PBS_NUM_NODES \* 4`
np=`expr $PBS_NUM_NODES / 4`
```

The command expr with “`” can be used to do integer math

For loops

```
myf90=`ls *f90`
for f in $myf90 ; do file $f ; done
doint.f90: ASCII program text
fourd.f90: ASCII program text
tintel.f90: ASCII program text
tp.f90: ASCII program text
vect.f90: ASCII program text
```

```
myf90=`ls *f90`
for f in $myf90
  do file $f
done
```

```
for (( c=1; c<=5; c++ )); do echo "Welcome $c times..."; done
Welcome 1 times...
Welcome 2 times...
Welcome 3 times...
Welcome 4 times...
Welcome 5 times...
```

```
for c in 1 2 3 4 5; do echo "Welcome $c times..."; done
Welcome 1 times...
Welcome 2 times...
Welcome 3 times...
Welcome 4 times...
Welcome 5 times...
```

```
for c in `seq 1 2 6`; do  echo "Welcome $c times..."; date; done
Welcome 1 times...
Tue Jul 31 12:17:11 MDT 2012
Welcome 3 times...
Tue Jul 31 12:17:11 MDT 2012
Welcome 5 times...
Tue Jul 31 12:17:11 MDT 2012
```

```
for c in `seq 1 2 6`
  do
  echo "Welcome $c
times..."
  date
done
```

"if" Test of Variable Being Set

We do this loop 3 times.

(1) "var" not set

(2) "var" set but empty

(3) var set and not empty

```
for i in 1 2 3 ; do
       echo "i=" $i
       if [ $i == 1 ] ; then unset var ; fi
       if [ $i == 2 ] ; then var="" ; fi
       if [ $i == 3 ] ; then var="abcd" ; fi

       if [ -z "$var" ]  ;     then echo "var is unset or empty A"; fi
       if [ ! -n "$var" ] ;    then echo "var is unset or empty A2"; fi
       if [ -z "${var-x}" ]  ; then echo "var is set but empty B"; fi
       if [ -n "$var" ] ;      then echo "var is set and not empty C"; fi
   echo
done
```

```
i= 1
var is unset or empty A
var is unset or empty A2

i= 2
var is unset or empty A
var is unset or empty A2
var is set but empty B

i= 3
var is set and not empty C
```

Combing Operations

Operation	Effect
[! EXPR]	True if **EXPR** is false.
[(EXPR)]	Returns the value of **EXPR**. This may be used to override the normal precedence of operators.
[EXPR1 -a EXPR2]	True if both **EXPR1** and **EXPR2** are true.
[EXPR1 -o EXPR2]	True if either **EXPR1** or **EXPR2** is true.

String Tests

```
if test "abc" = "def" ;then echo "abc = def" ; else echo "nope 1" ; fi

if test "abc" != "def" ;then echo "abc != def" ; else echo "nope 2" ; fi

if  [ "abc" \< "def" ];then echo "abc < def" ; else echo "nope 3" ; fi

if  [ "abc" \> "def" ]; then echo "abc > def" ; else echo "nope 4" ; fi

if  [ "abc" \> "abc" ]; then echo "abc > abc" ; else echo "nope 5" ; fi
```

```
nope 1
abc != def
abc < def
nope 4
nope 5
```

String Tests

```
if test "abc" = "def" ;then echo "abc = def" ; else echo "nope 1" ; fi
```

```
nope 1
```

```
if test "abc" != "def" ;then echo "abc != def" ; else echo "nope 2" ; fi
```

```
abc != def
```

```
if  [ "abc" \< "def" ];then echo "abc < def" ; else echo "nope 3" ; fi
```

```
abc < def
```

```
if  [ "abc" \> "def" ]; then echo "abc > def" ; else echo "nope 4" ; fi
```

```
nope 4
```

```
if  [ "abc" \> "abc" ]; then echo "abc > abc" ; else echo "nope 5" ; fi
```

```
nope 5
```

File Tests

Test	Meaning
[-a FILE]	True if FILE exists.
[-b FILE]	True if FILE exists and is a block-special file.
[-c FILE]	True if FILE exists and is a character-special file.
[-d FILE]	True if FILE exists and is a directory.
[-e FILE]	True if FILE exists.
[-f FILE]	True if FILE exists and is a regular file.
[-g FILE]	True if FILE exists and its SGID bit is set.
[-h FILE]	True if FILE exists and is a symbolic link.
[-k FILE]	True if FILE exists and its sticky bit is set.
[-p FILE]	True if FILE exists and is a named pipe (FIFO).
[-r FILE]	True if FILE exists and is readable.
[-s FILE]	True if FILE exists and has a size greater than zero.
[-t FD]	True if file descriptor FD is open and refers to a terminal.
[-u FILE]	True if FILE exists and its SUID (set user ID) bit is set.
[-w FILE]	True if FILE exists and is writable.
[-x FILE]	True if FILE exists and is executable.
[-O FILE]	True if FILE exists and is owned by the effective user ID.
[-G FILE]	True if FILE exists and is owned by the effective group ID.
[-L FILE]	True if FILE exists and is a symbolic link.
[-N FILE]	True if FILE exists and has been modified since it was last read.
[-S FILE]	True if FILE exists and is a socket.
[FILE1 -nt FILE2]	True if FILE1 has been changed more recently than FILE2, or if FILE1 exists and FILE2 does not.
[FILE1 -ot FILE2]	True if FILE1 is older than FILE2, or is FILE2 exists and FILE1 does not.
[FILE1 -ef FILE2]	True if FILE1 and FILE2 refer to the same device and inode numbers.

Checking Terminal Input

```
echo "Do you want to proceed?"
echo -n "Y/N: "
read yn
if [ $yn = "y" ] || [ $yn = "Y" ] ; then

  echo "You said yes"

else

  echo "You said no"
fi
```

Note spacing in the if statement. It is important!

Testing Return Code & /dev/null

- Commands return an exit code
  - 0 = success
  - not 0 = failure
- The exit code from the previous command is stored in $?
- $? can be echoed or tested
- This is often used with piping output into /dev/null "the bit bucket" when you only want to know if a command was successful

```
ls a_dummy_file >& /dev/null

if [ $? -eq 0 ] ; then
    echo "ls of a_dummy_file successful"
fi
```

While and with a Test and break

```
rm -f a_dummy_file
while true ; do
  ls a_dummy_file >& /dev/null
  if [ $? -eq 0 ] ; then
    echo "ls of a_dummy_file successful"
  else
    echo "ls of a_dummy_file failed"
  fi
  if [ -a a_dummy_file ] ; then
    echo "a_dummy_file exists, breaking"
    break
  else
    echo "a_dummy_file does not exist"
  fi
  touch a_dummy_file
  echo ; echo "bottom of while loop" ; echo
done
```

```
ls of a_dummy_file failed
a_dummy_file does not exist

bottom of while loop

ls of a_dummy_file successful
a_dummy_file exists, breaking
```

Command Line Arguments

```
[tkaiser@mio001 ~]$ ./cla word1 word2 word3
   echo $1 $2 $3
word1 word2 word3

   echo ${args[0]} ${args[1]} ${args[2]}
word1 word2 word3

   echo $@
word1 word2 word3  -> echo $@

   echo Number of arguments passed: $#
Number of arguments passed: 3

using for
for word1
for word2
for word3

using shift
word1
word2
word3
[tkaiser@mio001 ~]$
```

```
#!/bin/bash
# use predefined variables to access passed arguments
# echo arguments to the shell
echo '    echo $1 $2 $3'
echo $1 $2 $3
echo
# We can also store arguments from bash command line in special array
args=("$@")
#echo arguments to the shell
echo '    echo ${args[0]} ${args[1]} ${args[2]}'
echo ${args[0]} ${args[1]} ${args[2]}
echo
#use $@ to print out all arguments at once
echo '    echo $@'
echo $@ ' -> echo $@'
echo
# use $# variable to print out
# number of arguments passed to the bash script
echo '    echo Number of arguments passed: $# '
echo Number of arguments passed: $#
echo

echo using for
for a in $@ ; do
  echo "for" $a
done
echo
#this prints all arguments
echo using shift
while test $# -gt 0
do
    echo $1
    shift
done
```

A Script to “tail” a set of files

```
#!/bin/bash

# first argument is the number of lines to show
n=$1
shift

# shift discards it now we loop over the rest

for a in $@ ; do
# test to see if it is a regular file
        if [ -f $a ]; then
                echo "**** "  $a " ****”
# run tail “else” say it is not a regular FILE
                tail -n $n $a
        else echo $a not a regular FILE
        fi
done
echo
```

Random Stuff

- Sed examples
- Awk examples
- Sort examples

Some examples

`sed "s/\..*//"` Remove everything after a period

`sed "s/.*<r//"` Remove everything before <r

`awk '{print $NF}'` Print the last item on every line of a file

```
[tkaiser@mc2 h2ogs]$ grep real_time set* | sed "s/set//" | sort  -t_ -k1n,1 -k3n,3 -k2n,2
01_1_1:<real_time> 87.70 </real_time>
01_2_1:<real_time> 45.40 </real_time>
01_4_1:<real_time> 25.04 </real_time>
01_8_1:<real_time> 14.83 </real_time>
01_16_1:<real_time> 9.62 </real_time>
…
…
04_4_16:<real_time> 5.78 </real_time>
04_1_32:<real_time> 15.66 </real_time>
04_2_32:<real_time> 9.81 </real_time>
04_1_64:<real_time> 25.84 </real_time>
[tkaiser@mc2 h2ogs]$
```

Remove “set” from each line
sort numerically by fields 1 then 3 then 2
with _ as the delimiter between fields

Note: sed treats the first character after 's' as the separator in the search-replace function. By tradition a “/“ is used. If you are working with lists of filenames or html tags that contain “/“ another character such as “#” or “,” can be used as the separator

Linux links

- Tutorials:
  - http://www.ee.surrey.ac.uk/Teaching/Unix/
  - https://www.cac.cornell.edu/VW/Linux/default.aspx?id=xup_guest
  - http://tille.garrels.be/training/bash/
  - See: http://geco.mines.edu/scripts/
- General Interest
  - http://en.wikipedia.org/wiki/History_of_Linux
  - http://en.wikipedia.org/wiki/Linux_distribution

Local ssh pages

- Setting up ssh, including putty
  - http://geco.mines.edu/ssh/
- Tunneling
  - http://geco.mines.edu/ssh/tunneling.html
  - http://hpc.mines.edu/bluem/transfer.html#scp
  - http://hpc.mines.edu/bluem/multistage.html

More Links

- Home Page
  - hpc.mines.edu
- Blog
  - http://geco.mines.edu/hpcbook.shtml
- BlueM
  - http://hpc.mines.edu/bluem/
- Mio
  - http://inside.mines.edu/mio/

More Links

- BlueM Load
  - http://mindy.mines.edu
- Module links:
  - http://inside.mines.edu/mio/mio001/mod.html
  - http://mindy.mines.edu/modules/aun/
  - http://mindy.mines.edu/modules/mc2/