Introduction to UNIX and Perl for Biologists

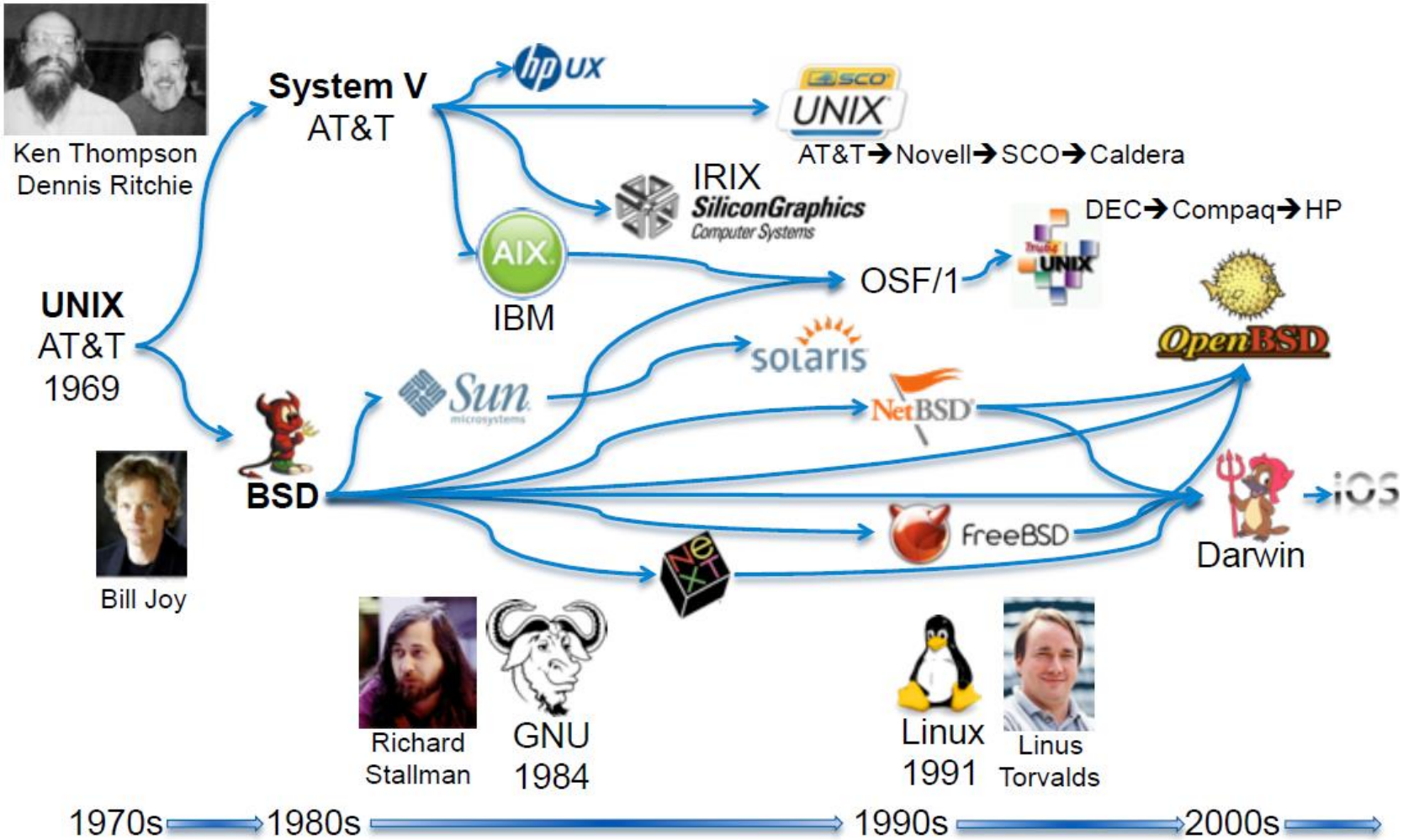
Adam Burkholder

burkholderab@mail.nih.gov

National Institute of Environmental Health Sciences

Integrative Bioinformatics

What is UNIX?



Darrell Hurt, NIAID

Why Use UNIX?

- Files unmanageably large
- Analysis tools only available for UNIX
- Only way to access high performance computing resources (HPC) at NIEHS

HPC Resources at NIEHS

- Server (wine)
  - 32 CPU's
  - 512 GB Memory
- Storage
  - 550 TB
  - Parallel File System
- Designed to be shared by a large number of simultaneous users

How to Access HPC Resources

- An account is created for you when you first receive nextgen sequencing data from NISC
- Send an email to NIEHS Scientific Computing Support (scs@niehs.nih.gov)
  - Request an account on the wine server
  - Include your NIH username and Lab Group

Important Notes

- All file and command names are case sensitive
  - "test.txt" and "TEST.txt" can be two separate files stored in the same directory
- Using spaces in file names is discouraged
  - Use underscores or CamelCase instead (e.g. "file_name" or "fileName" instead of "file name")
- There is no "undo"
  - Files overwritten or modified cannot be restored to their original state
- There is no "Trash" or "Recycle Bin"
  - File deletion is permanent

The UNIX Shell (bash)

- Primary user interface for UNIX operating systems
- Allows editing of commands
- Autocompletes command or file names (tab)
- Allows use of wildcard character (*) to match one or more file names
- Manages command history
  - Use up and down arrows to cycle through previously run commands

Navigating the UNIX File System

- pwd – Prints the current "working" directory
  - "/" is the root directory, similar to C: in Windows
  - UNIX systems use forward slashes "/" rather than back slashes to separate directory names
  - Your "home directory" can also be accessed using the Finder
- mkdir unix_class – Creates a new directory called "unix_class"
  - The name of the directory is passed to mkdir as an argument, commands may take zero arguments, such as pwd, or many
- cd unix_class – Changes the current directory to "unix_class"
- cd .. – Shifts the current directory one level up
  - ".." always refers to the directory one level above the current directory
  - "." always refers to the current directory
- cd unix_class – Return to the "unix_class" directory

File Management

- cp ../Desktop/unix_class.tar.gz . – Copies the file to the current directory
- ls – Lists the files in the current directory
- cp unix_class.tar.gz temp.tar.gz – Makes a copy of the file named "temp.tar.gz"
- mkdir tempdir
- mv temp.tar.gz tempdir – Move temp.tar.gz to the new directory
- cd tempdir
- mv temp.tar.gz temporary.tar.gz – Change the name of temp.tar.gz to temporary.tar.gz

File Management

- cd ..
- ls -l – Lists the files in the current directory with additional information
  - "-l" is an option or parameter passed to ls that modifies its behavior
  - Other useful parameters for ls include:
    - -t: Sort by date modified
    - -S: Sort by file size
    - -R: List files in subdirectories as well
  - Output includes file permissions, file owner, file group, file size, and date modified

File Permissions

d $\underbrace{\text{rwx}}_{\text{Owner}}$ $\underbrace{\text{r-x}}_{\text{Group}}$ $\underbrace{\text{- - -}}_{\text{Others}}$

- r – Read Files/List Files in a Directory
- w – Modify Files/Write New Files to a Directory
- x – Execute a Program or Script/Run Commands within a Directory

Deleting Files

- cd tempdir
- rm temporary.tar.gz – Delete the file
  - There is no "Trash" or "Recycle Bin" on UNIX systems, file deleted using rm are deleted permanently
- cd ..
- rmdir tempdir – Delete the temporary directory
  - Directories must be empty prior to deletion

Working with Archives

- gunzip unix_class.tar.gz – Decompresses the "tarball" to "unix_class.tar"
  - "unzip" decompresses ".zip" files
- tar -xvf unix_class.tar – Unpacks the files contained in "unix_class.tar"
  - x, v, and f are separate parameters, equivalent to -x -v -f unix_class.tar
  - x indicates files are to be extracted from the archive, v tells tar to list the files as they are unpacked, and f is used to tell tar the name of the file to unpack

Working with Text Files

- cat sample1_peaks.bed – Prints entire file to shell
- cat sample1_peaks.bed|more – Prints the file, one screen at a time
  - The pipe (|) indicates that the output of "cat" should be directed to the input of "more"
  - Scroll with up and down arrows, jump one screen at a time with space bar, quit by pressing "q"
- head sample1_peaks.bed – Prints the first 10 lines of file
- head -n 100 sample1_peaks.bed – Prints the first 100 lines of file
- tail sample1_peaks.bed – Prints the last 10 lines of file
- head -n 100 sample1_peaks.bed|tail -n 1 – Prints line 100 only

Working with Text Files

- cat sample1_peaks.bed > temp.bed – Prints the entire files and stores the output in temp.bed, essentially copying the file
  - Similar to the pipe, the ">" indicates that the output of "cat" should be directed to the file "temp.bed"
- diff sample1_peaks.bed temp.bed – Compares the contents of two files
- cat sample1_peaks.bed >> temp.bed – Appends the output of "cat" to temp.bed, rather than overwriting it
- diff sample1_peaks.bed temp.bed
  - The files are no longer identical
- sort -k1,1 -k2,2n temp.bed > sort.bed – Sorts temp.bed by the first column alphabetically, the by the second column numerically

Managing Jobs

- Use Ctrl-C to Interrupt a Job
- Use Ctrl-Z to Suspend a Job
- A suspended job can be restarted by running "fg"
- A suspended job can be placed in the background by running "bg"
  - Additional commands can be run while a job is running in the background

Managing Jobs

- Append a "&" to a command to start it in the background
- Use "ps" to view current jobs
  - All jobs have unique process ID's, displayed in the first column of the output
- Use "kill" followed by the process ID to end problematic jobs
  - Example: kill 5109
- Use "top" to view all jobs running on the server
  - Quit top with “q”

Working with Text Files

- uniq sort.bed > unique.bed – Removes duplicate lines from sort.bed
- diff sample1_peaks.bed unique.bed – The files are now identical
- wc -l *.bed – Count the number of lines in each BED file
- cut -f 1 sample1_peaks.bed > sample1_peaks_chrs.txt – Print only the first field of each line
- cut -f 2 sample1_peaks.bed > sample1_peaks_starts.txt
- paste sample1_peaks_chrs.txt sample1_peaks_starts.txt sample1_peaks_starts.txt > joined.txt – Join the extracted columns into a new three column BED file containing peak starts rather than intervals

Working with Text Files

- grep "chr3" sample1_peaks.bed – Search through the BED file, print all lines that contain "chr3"
  - grep is a very fast and powerful search tool that is highly configurable
- man grep – Displays the manual page for the grep command

Practical Examples

- Merge Two BED Files, Retaining Only Unique Peak Locations:
  - cat sample*_peaks.bed|sort -k1,1 -k2,2n|uniq > merged.bed &
- Find Peak Locations Common to Two BED Files:
  - cat sample*_peaks.bed|sort -k1,1 –k2,2n|uniq -d > common.bed
  - wc –l *.bed

Wrapping Up

- history > unix_class.txt – Stores a log of all the commands run to the file unix_class.txt
- exit – Log out of the server

Server Etiquette

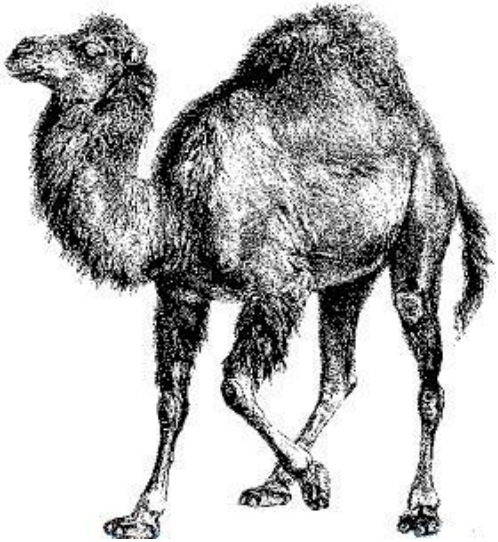
- When you first log in
  - Run "who" to see how crowded the server is
  - Run "top" to see what jobs are currently running
- Do not use an excessive number of CPU's at one time
- Keep an eye on the amount of memory your jobs are using

Storage Organization

- Home directories can be found in /ddn/gs1/home
- Lab Group directories can be found in /ddn/gs1/project/nextgen/post
  - Nextgen sequencing data stored in this location
- Analysis tools can be found in /biotools
  - If an open source tool you need has not been installed, contact NIEHS Scientific Computing Support (scs@niehs.nih.gov)

What is Perl?

- General purpose programming/scripting language
- Very useful and fast for processing text
- Can be run as a script saved in a file, or directly from the command line
- Typically pre-installed in UNIX systems



One Liners

- perl -e '[*perl code*]'
- Examples:
  - perl -e 'print "hello world\n";'
  - perl -e '$sum=10*((12/3)+2); print "$sum\n";'
- Additional Parameters:
  - n: Loop through file line by line
  - p: Loop through file line by line and print
  - a: Split each line by whitespace
  - F'[string]': Split line by character rather than whitespace

Perl Variables

- Scalars
  - Names begin with $
  - Hold numbers or strings
- Arrays/Lists
  - Names begin with @
  - Hold multiple scalars
- Hashes
  - Names begin with %
  - Hold multiple pairs of scalars

Useful One Liner Template

- perl -ane 'print "$F[0]\t$F[1]\t...$F[n]\n";' in > out
- Special Variable
  - "@F" stores the current line separated into fields or columns by whitespace
    - $F[0], $F[1], $F[2], etc. allow access to individual fields
- Print only selected fields from a multi-column input file, rearrange columns
  - Can be used similarly to "cut" and "paste", without the need for intermediate files
  - perl -ane 'print "$F[0]\t$F[1]\t$F[1]\n";' sample1_peaks.bed > sample1_points.bed

The if Statement

- if(*condition*) { *perl code* }: only run the code if the condition is true
- Test Operators
  - eq|ne: equal and not equal, for use with strings
  - ==|!=: equal and not equal, for use with numbers
  - <|>|<=|>=: less than, greater than, less than or equal to, greater than or equal to
- else { *perl code* }: only run the code if the condition is false
- Example
  - perl -e 'if(1==2) { print "yes\n"; } else { print "no\n"; }'

Useful One Liner Template

- perl -ane 'if(X) { print Y; } else { print Z; }' in > out
- Prints different output depending on the outcome of condition "X"
- Special Variable
  - "$_" stores the current line when looping through a file
- Example: extract all peaks intervals, 200 nt or shorter, from a BED file
  - perl -ane 'if(($F[2]-$F[1])<=200) { print $_; }' sample1_peaks.bed > sample1_peaks_max200nt.bed

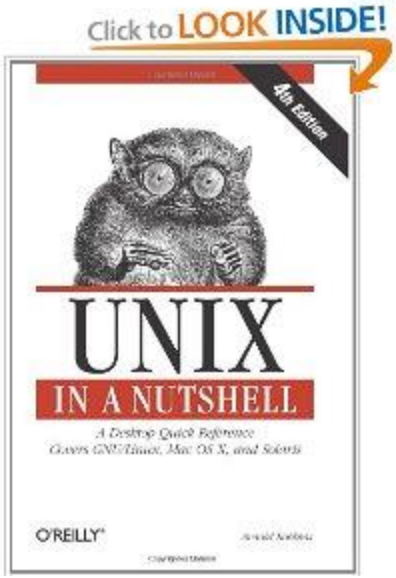
BEGIN, END, and foreach

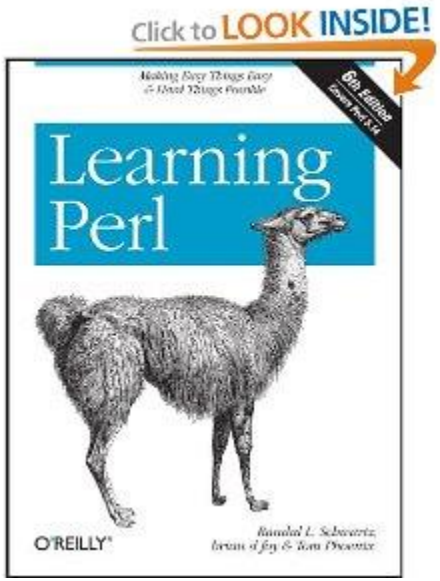
- BEGIN { *perl code* }: run this code once before the first line of the file is read
- END { *perl code* }: run this code once after the last line of the file is read
- foreach(*@array*) { *perl code* }: run the code on each element stored in the array
  - $_ stores the current array element

Useful One Liner Template

- perl -ane '$hash{$key}++;
  END {
  foreach(keys(%hash)) {
  print "$_\t$hash{$_}\n";
  }
  }' in > out
- Counts the number of times each "$key" is encountered, then prints the results
- Example: Determine the distribution of peak sizes in a BED file
  - perl -ane '$table{$F[2]-$F[1]}++;
    END {
    foreach(keys(%table)) {
    print "$_\t$table{$_}\n";
    }
    }' sample1_peaks.bed|sort -k1,1n > sample1_peaks_distribution.txt

Additional Resources

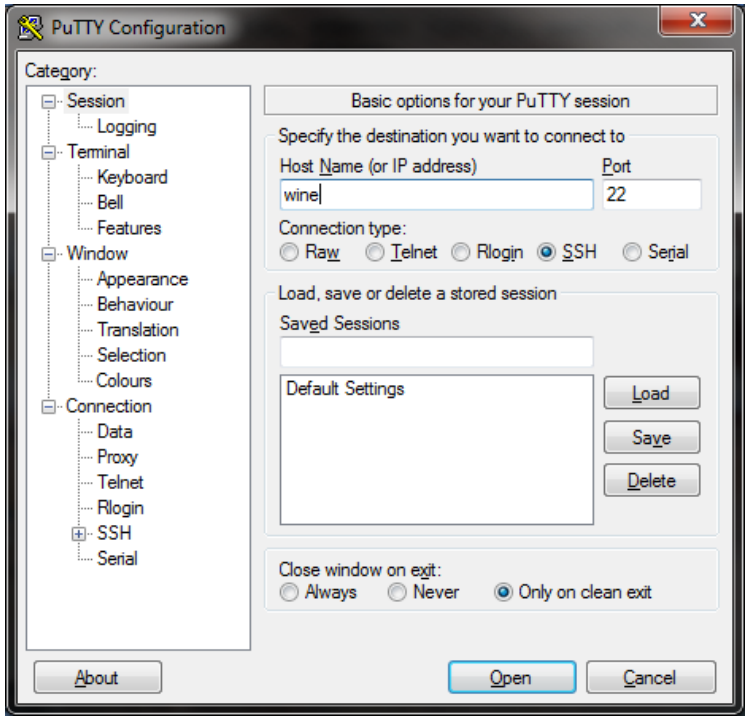




Appendix

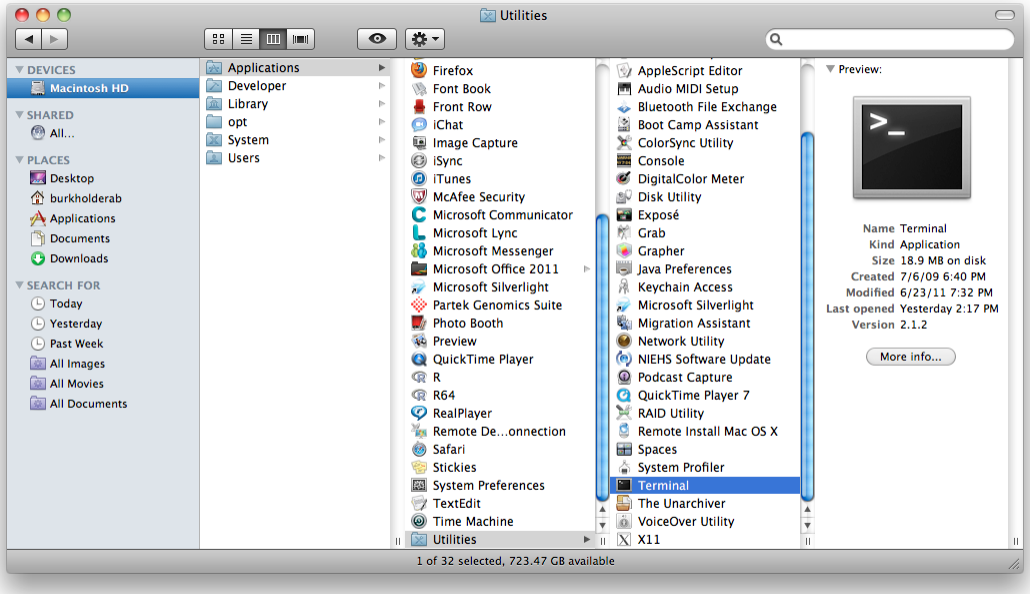
Logging into wine (Windows)

- Download puTTY SSH client, and save to your Desktop
  - http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe
- Leave all settings at their default values
- Type "wine" into the Host Name field
- Click Open



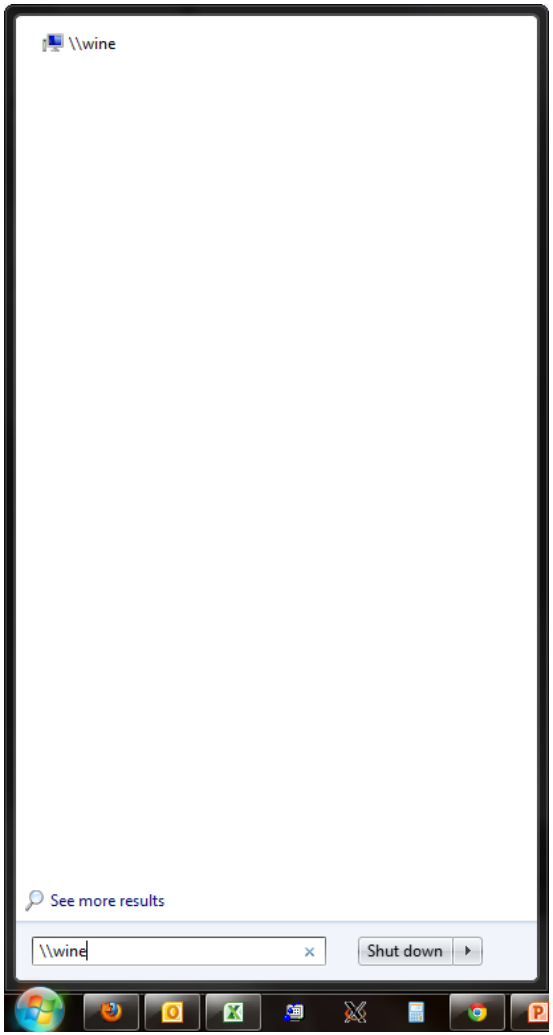
Logging into wine (Mac)

- In the Finder, navigate to Macintosh HD -> Applications -> Utilities
- Drag "Terminal" to the Dock and Click to Run
- Type "ssh wine"
- When prompted, type "yes"



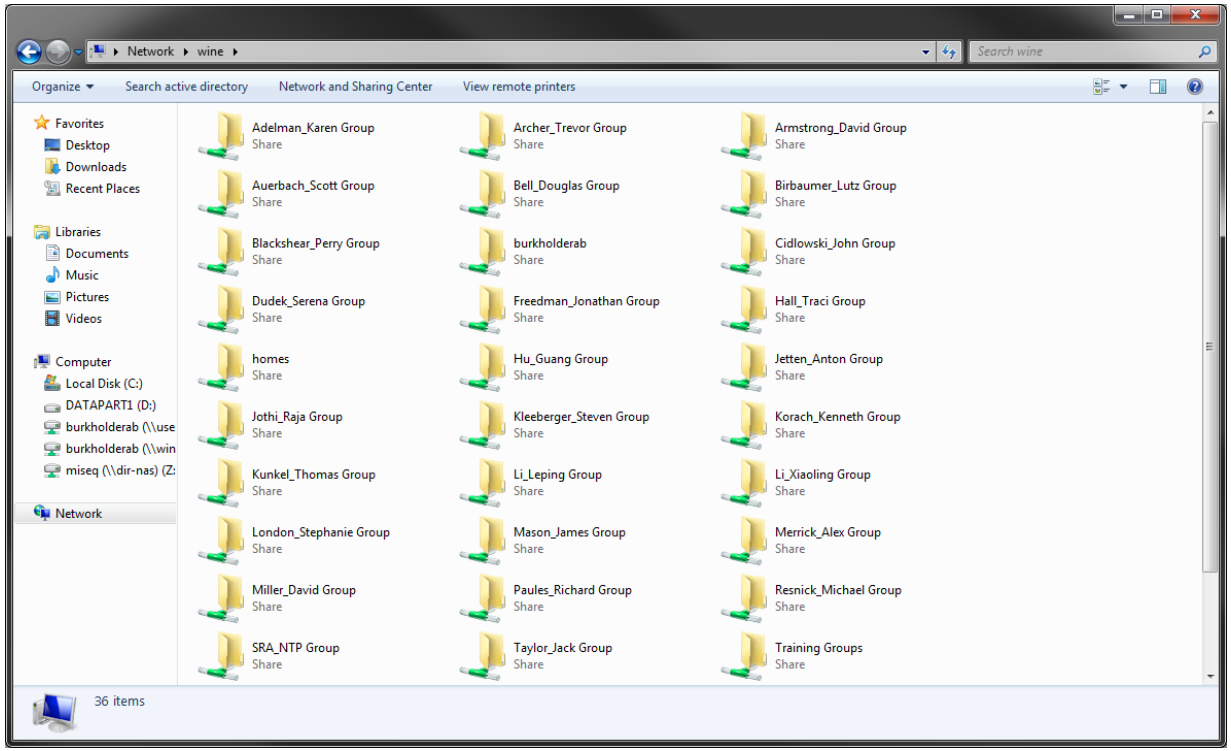
Transferring Files (Windows)

- Open the Start Menu
- Type \\wine in the "Search programs and files" field



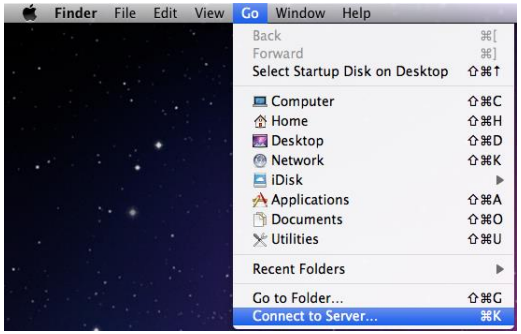
Transferring Files (Windows)

- Create, Delete, Copy, and Move files as you normally would

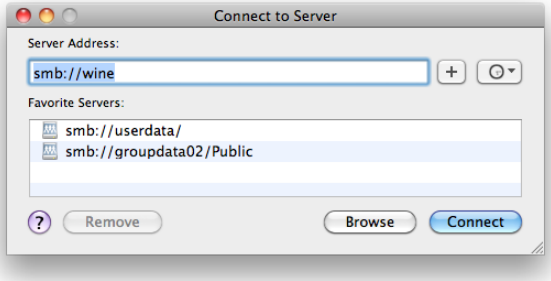


Transferring Files (Mac)

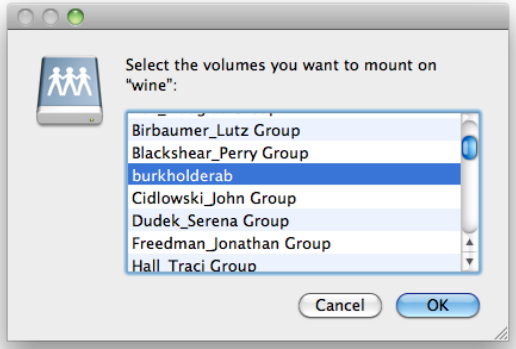
- Select "Connect to Server" from "Go" menu



- Type "smb://wine" into the Server Address field, click Connect



- Select a folder to mount, Create, Copy, Delete, and Move files normally



Changing File Permissions

- File permissions can be modified using "chmod"
- Example
  - mkdir temp
  - chmod u-w temp – Disables user (owner) write permissions in tempdir
    - "u" refers to owner permissions, "g" refers to group permissions, "o" refers to other, and "a" refers to all three
    - To enable permissions, use "+" instead of "-"
  - ls > temp/test.txt
    - Results in Permission Denied error message
  - chmod u+w temp – Re-enables owner write permissions
  - ls > temp/test.txt
    - Results in creation of test.txt

Managing Jobs

- Run jobs expected to take a very long time using "nohup"
  - This allows the job to run even if you log out of the server
  - Example: nohup grep "chr3" sample1_peaks.bed > sample1_peaks_chr3.bed
- Use "disown" to "nohup" a job after it has been started