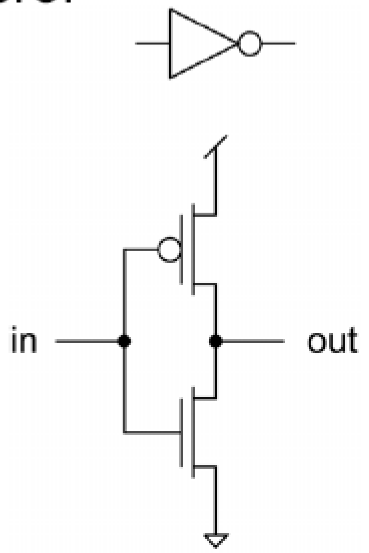
Biologists at the computer

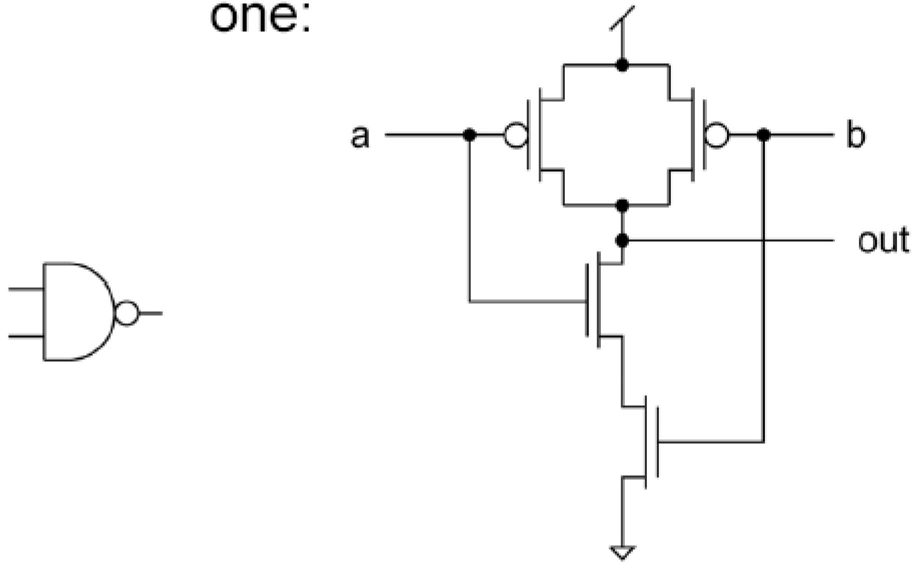
HMS
Leon Peshkin
pesha@hms.harvard.edu

Transistor-level Logic Circuits

Simple rule for wiring up MOSFETs:

- nFET is used only to pass logic zero:
- pFet is used only to pass logic one:



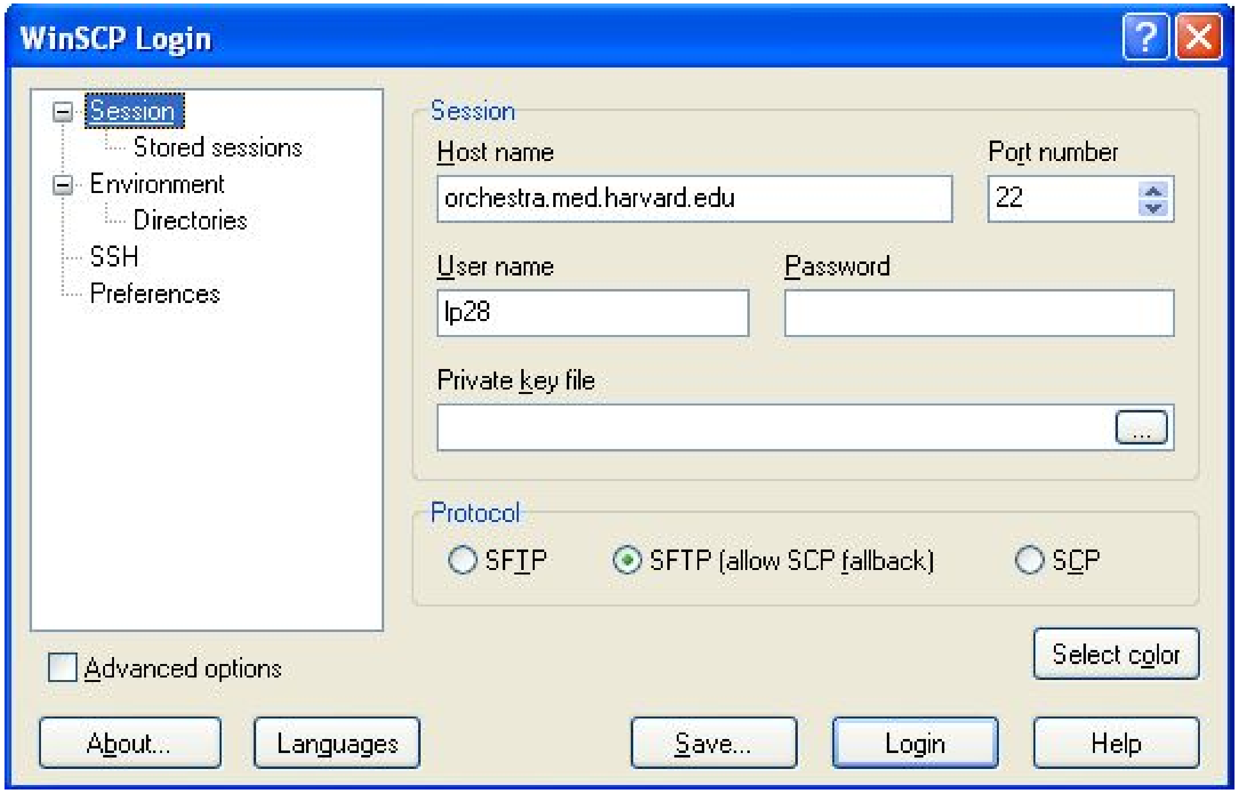


Note: This rule is sometimes violated by expert designers under special conditions.

Setup

Putty

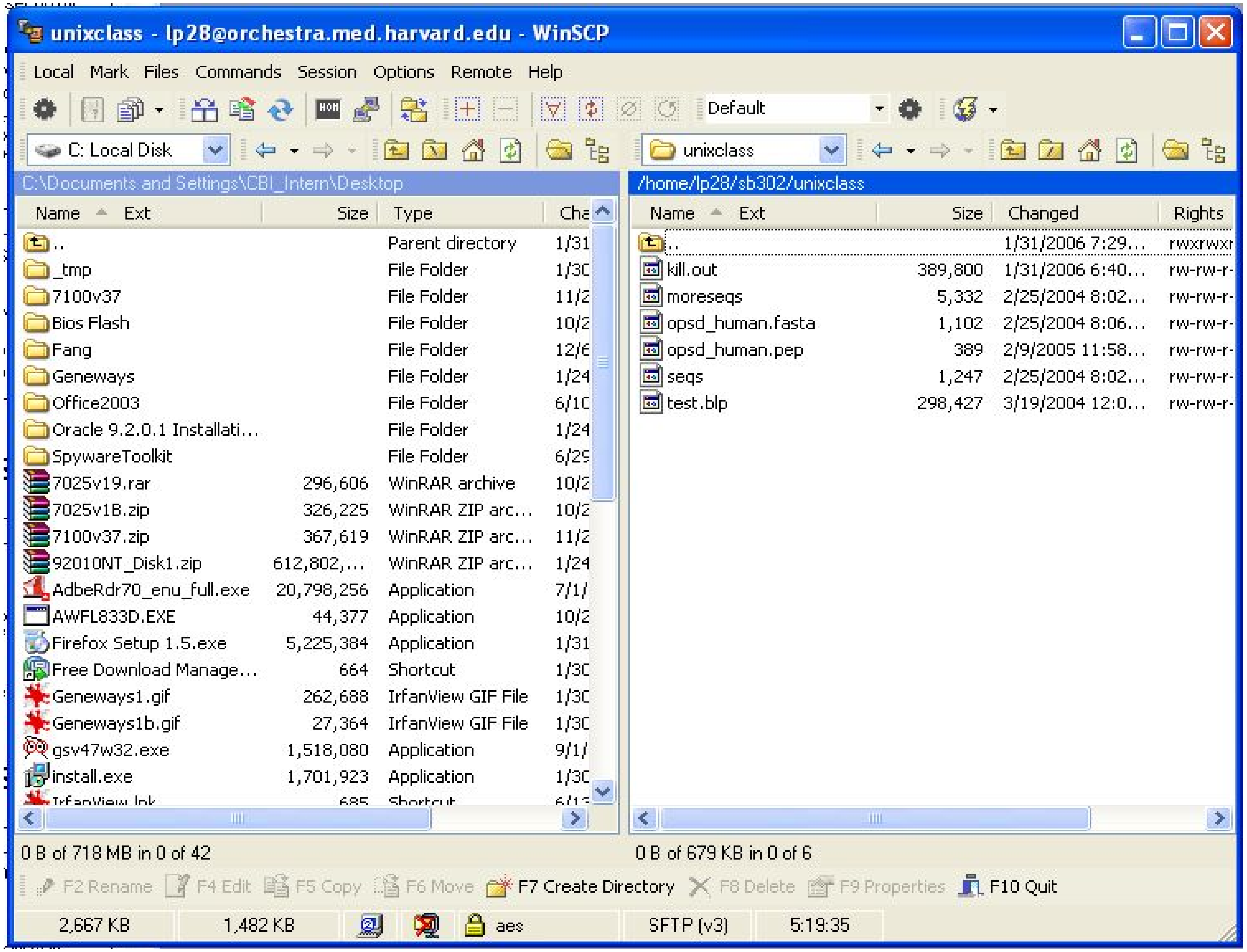
WinSCP


WinSCP Login

Session
Host name: orchestra.med.harvard.edu
Port number: 22
User name: lp28
Password:
Private key file:

Protocol
SFTP [✓ SFTP (allow SCP fallback)] SCP

http://cbi.med.harvard.edu/people/peshkin/sb302/fragments1.zip



unixclass - lp28@orchestra.med.harvard.edu - WinSCP

C:\Documents and Settings\CBI_Intern\Desktop

Name	Ext	Size	Type	Cha
..			Parent directory	1/31
_tmp			File Folder	1/30
7100v37			File Folder	11/2
Bios Flash			File Folder	10/2
Fang			File Folder	12/6
Geneways			File Folder	1/24
Office2003			File Folder	6/10
Oracle 9.2.0.1 Installati…			File Folder	1/24
SpywareToolkit			File Folder	6/29
7025v19.rar		296,606	WinRAR archive	10/2
7025v1B.zip		326,225	WinRAR ZIP arc…	10/2
7100v37.zip		367,619	WinRAR ZIP arc…	11/2
92010NT_Disk1.zip		612,802,…	WinRAR ZIP arc…	1/24
AdbeRdr70_enu_full.exe		20,798,256	Application	7/1/
AWFL833D.EXE		44,377	Application	10/2
Firefox Setup 1.5.exe		5,225,384	Application	1/31
Free Download Manage…		664	Shortcut	1/30
Geneways1.gif		262,688	IrfanView GIF File	1/30
Geneways1b.gif		27,364	IrfanView GIF File	1/30
gsv47w32.exe		1,518,080	Application	9/1/
install.exe		1,701,923	Application	1/30
IrfanView.lnk		685	Shortcut	6/1

/home/lp28/sb302/unixclass

Name	Ext	Size	Changed	Rights
..			1/31/2006 7:29…	rwxrwxr
kill.out		389,800	1/31/2006 6:40…	rw-rw-r-
moreseqs		5,332	2/25/2004 8:02…	rw-rw-r-
opsd_human.fasta		1,102	2/25/2004 8:06…	rw-rw-r-
opsd_human.pep		389	2/9/2005 11:58…	rw-rw-r-
seqs		1,247	2/25/2004 8:02…	rw-rw-r-
test.blp		298,427	3/19/2004 12:0…	rw-rw-r-

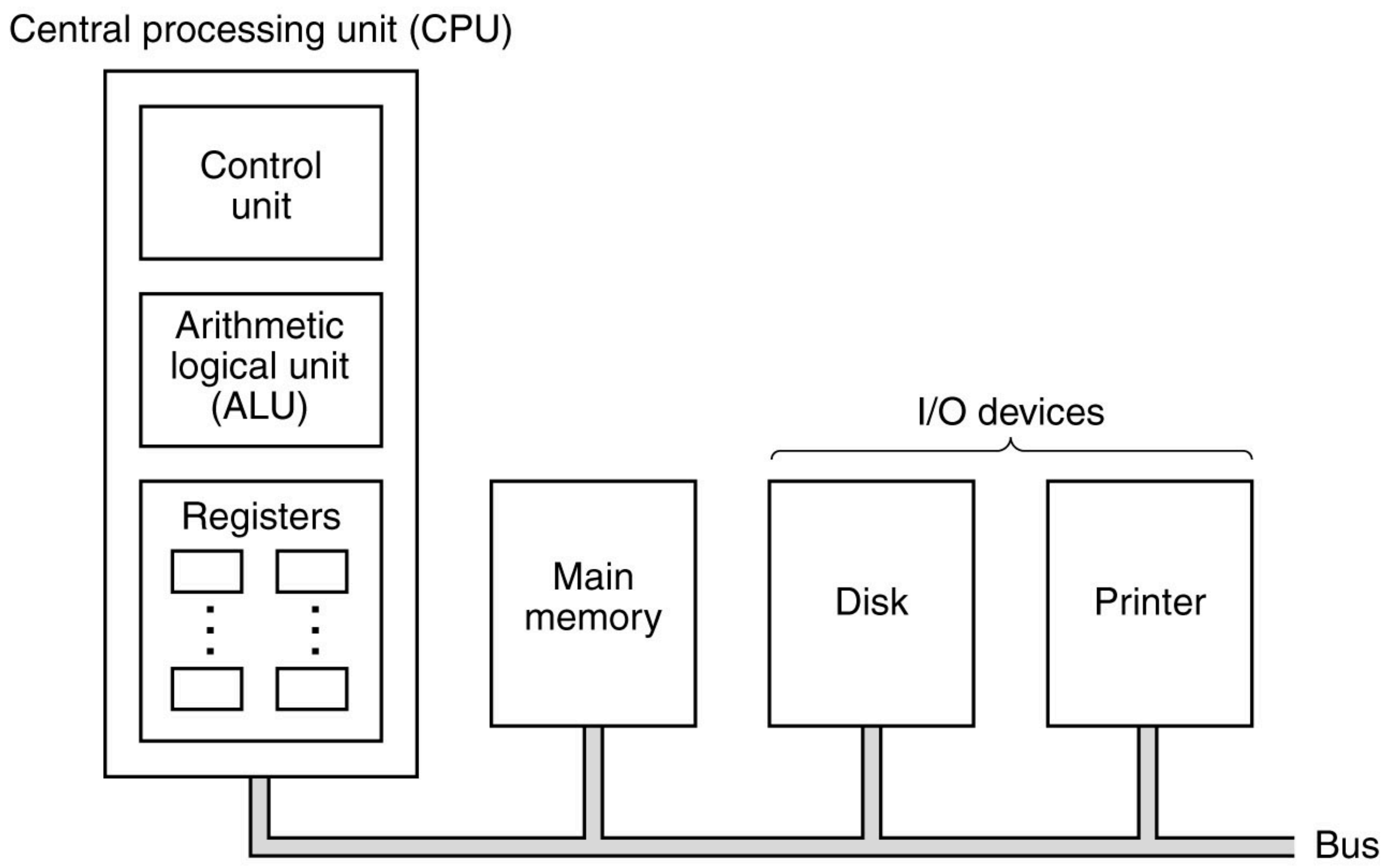
0 B of 718 MB in 0 of 42

0 B of 679 KB in 0 of 6

2,667 KB | 1,482 KB | aes | SFTP (v3) | 5:19:35

Anatomy

- Abstraction data/code/image/music
- Modularity = objects = client/server



Introduction to UNIX/Linux & the Orchestra Cluster

- Become familiar with UNIX/Linux OS
- Manipulate files and folders
- Run bioinformatics programs from the “command line”
- Running jobs on Linux Cluster

Outline

- Getting Started: What's Unix?
- Getting In: Logging into Unix
- Getting Stuff Done: Commands
- Getting Around: The Filesystem
- Getting Fancy: Complex Commands
- Getting CPU Time: Using the Cluster

UNIX/Linux: What is it?

- UNIX is an <u>operating system (OS)</u>
  - An OS is a set of files/programs that control and organize the resources of a computer.
- UNIX comes in many flavors and runs on many different architectures (types of computer hardware).
- UNIX is called an <u>interactive timesharing system</u>.
- Linux is a kind of UNIX.

Examples of OSs (winxp, mac osx, linux, solaris)

Why use it?

- Many core bioinformatics tools were developed for UNIX (BLAST, PHRAP, GCG/EMBOSS, HMMer, etc.)
- UNIX is a multi-user, multitasking, robust OS designed for networking.
- Excellent programming tools available that can be implemented without developing a GUI (Graphical User Interface).
- Widely used and many open source projects exist

"*Good composers borrow; great composers steal.*"

-Igor Stravinsky

UNIX Flavors

-Commercial

- Solaris (Sun Microsystems)
- AIX (IBM)
- HP-UX (Hewlett Packard)
- Tru64 Unix (Compaq/HP)
- Mac OS X (Apple)

-Open Source (~Free)

- FreeBSD
- BSD/Other – Darwin/NetBSD (Intel, PowerPC)
- **Linux** (Intel, Alpha, Sun Sparc, PowerPC, ARM, Amiga)
  - There are many kinds of Linux (which some call GNU/Linux)
  - We use RedHat

You will (usually) need to download a separate program (binary) to run on Windows, Mac, and each kind of UNIX.

Outline

- Getting Started: What's Unix?
- Getting In: Logging into Unix
- Getting Stuff Done: Commands
- Getting Around: The Filesystem
- Getting Fancy: Complex Commands
- Getting CPU Time: Using the Cluster

How to connect or “login”

- You need your login name (user name) and password.
- Generally, UNIX account administrators will give you an initial password
  - change it when you first login
- Many people can be logged in concurrently (multi-user)
- People can run many jobs concurrently (multi-tasking)
- orchestra, the RITG cluster, requires secure connections
  - Use a program that supports “SSH”, the *secure shell protocol* (which encrypts data flow between computers)

SSH Clients

WIN

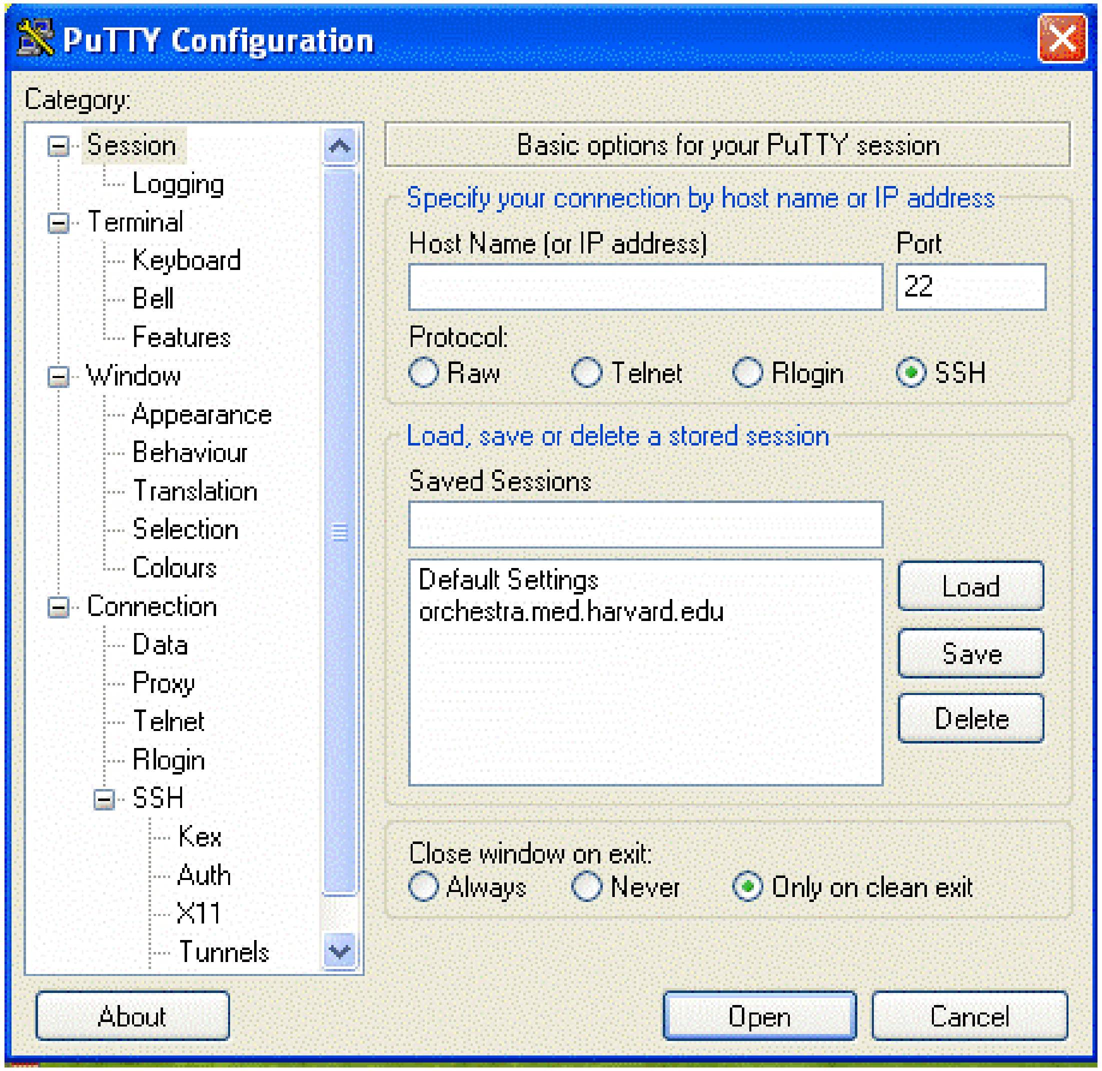
- Putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/)
- Teraterm (http://hp.vector.co.jp/authors/VA002416/teraterm.html)
- SecureCRT (http://www.fas.harvard.edu/cgi-bin/software/download.pl)

MAC

- Terminal.app (Apple OSX)
- BetterTelnet (http://www.cstone.net/~rbraun/mac/telnet/)
- NiftyTelnet (http://www.niftytelnet.org)

The SSH client

- Putty.exe
- Icon on the desktop
- Double click to launch
- Type in hostname and select port OR choose from Saved Sessions



PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify your connection by host name or IP address

Host Name (or IP address):

Port: 22

Protocol: Raw Telnet Rlogin [✓ SSH]

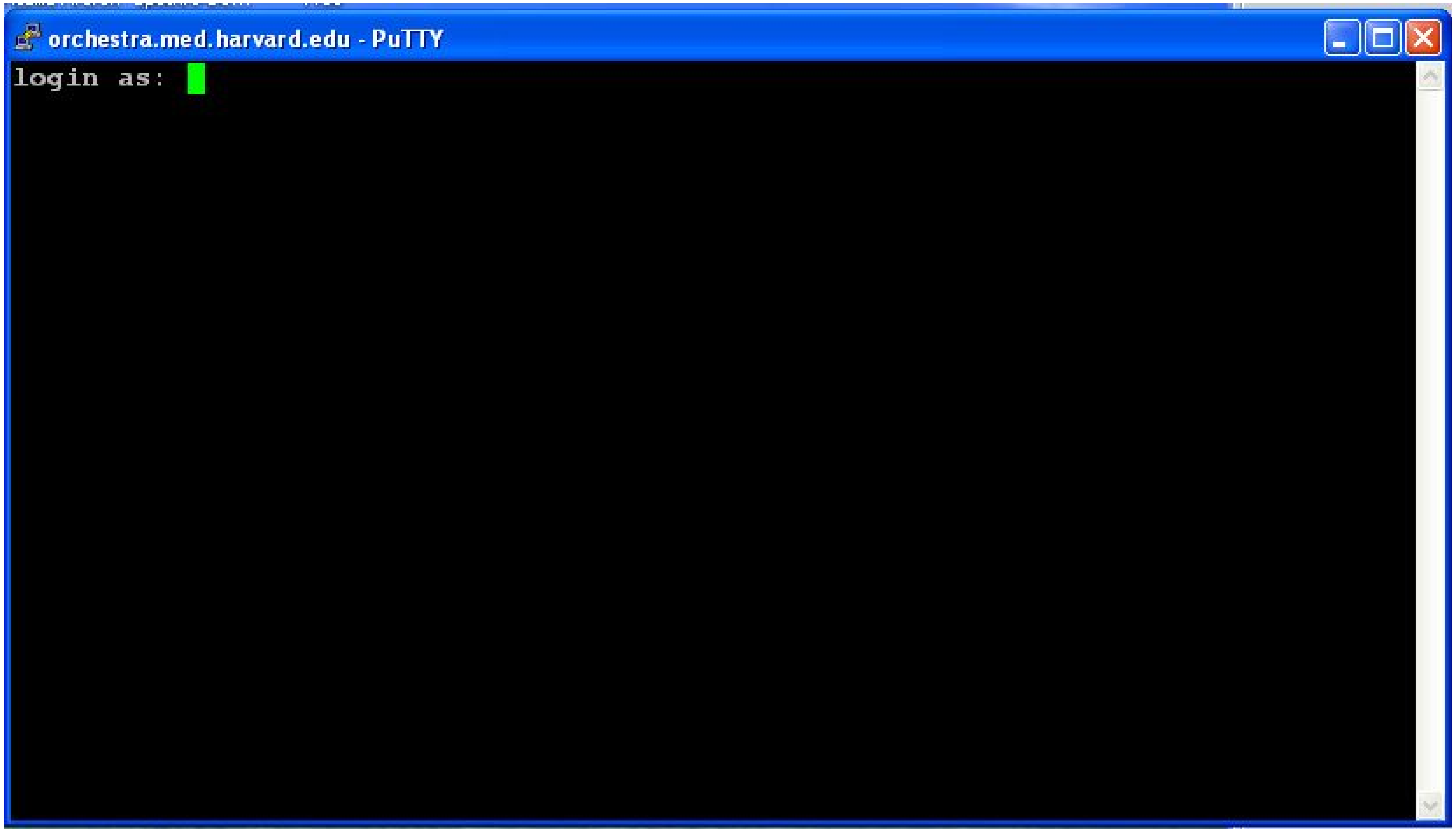
Load, save or delete a stored session

Saved Sessions

Default Settings
orchestra.med.harvard.edu

Close window on exit: Always Never [✓ Only on clean exit]

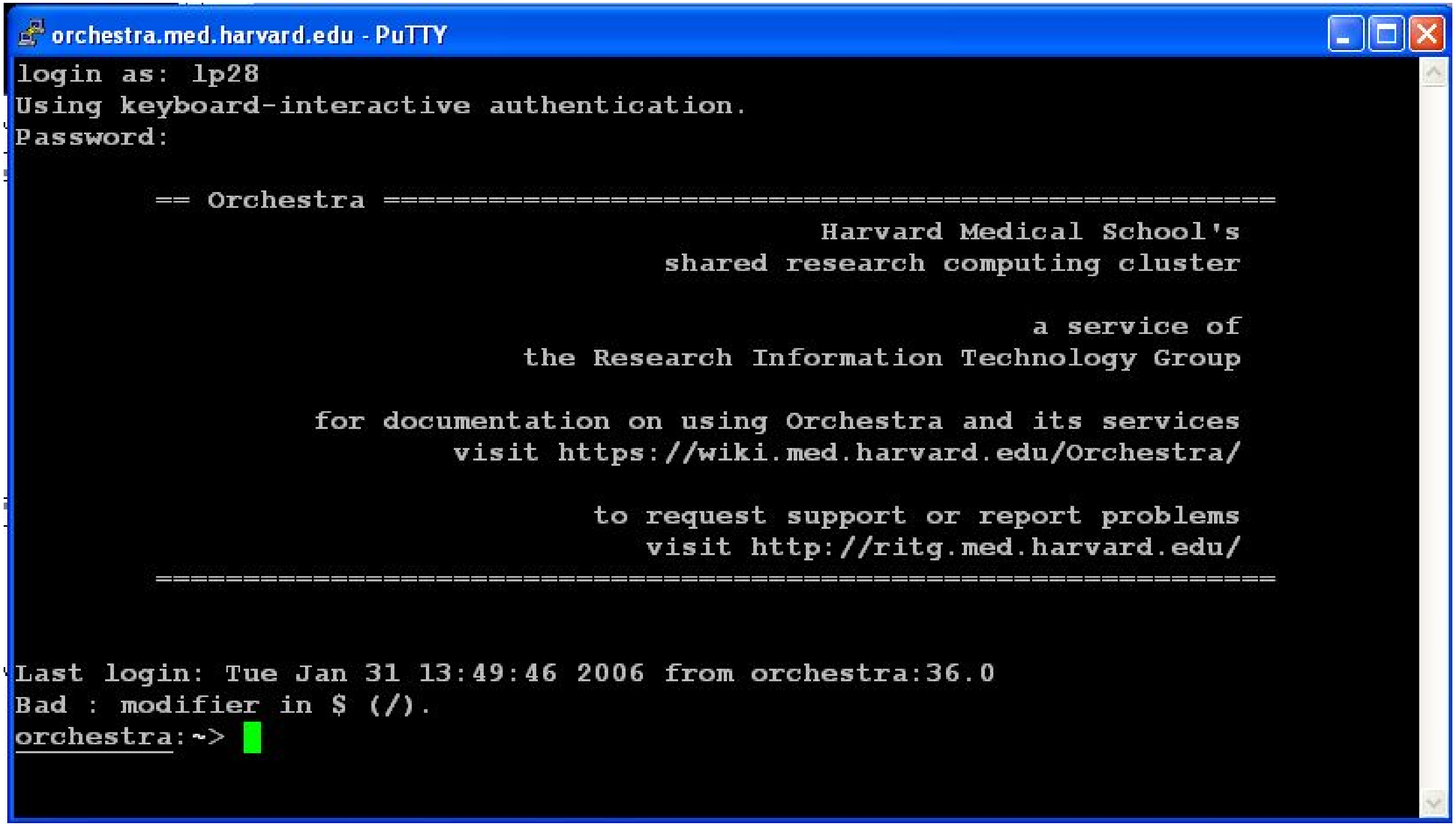
Terminal Window



```
login as:
```

From here on, it's the same whether you use SSH, another program, or a monitor directly connected to a UNIX machine

Terminal login



orchestra.med.harvard.edu - PuTTY

```
login as:  lp28
Using keyboard-interactive authentication.
Password:

        == Orchestra =====================================================
                                              Harvard Medical School's
                                     shared research computing cluster

                                                          a service of
                             the Research Information Technology Group

                 for documentation on using Orchestra and its services
                         visit https://wiki.med.harvard.edu/Orchestra/

                                     to request support or report problems
                                        visit http://ritg.med.harvard.edu/
        ==================================================================



Last login: Tue Jan 31 13:49:46 2006 from orchestra:36.0
Bad : modifier in $ (/).
orchestra:~>
```

Logging in: 1st Time

- Connect to

```
ssh orchestra.med.harvard.edu
```

- To change your password (*we won't do this today*):

```
passwd
```

- To logout:

```
logout/exit
```

Exercise 1 - Logging in

- Login to portal
- Don't logout - we have much more to do!

Outline

- Getting Started: What's Unix?
- Getting In: Logging into Unix
- Getting Stuff Done: Commands
- Getting Around: The Filesystem
- Getting Fancy: Complex Commands
- Getting CPU Time: Using the Cluster

Commands

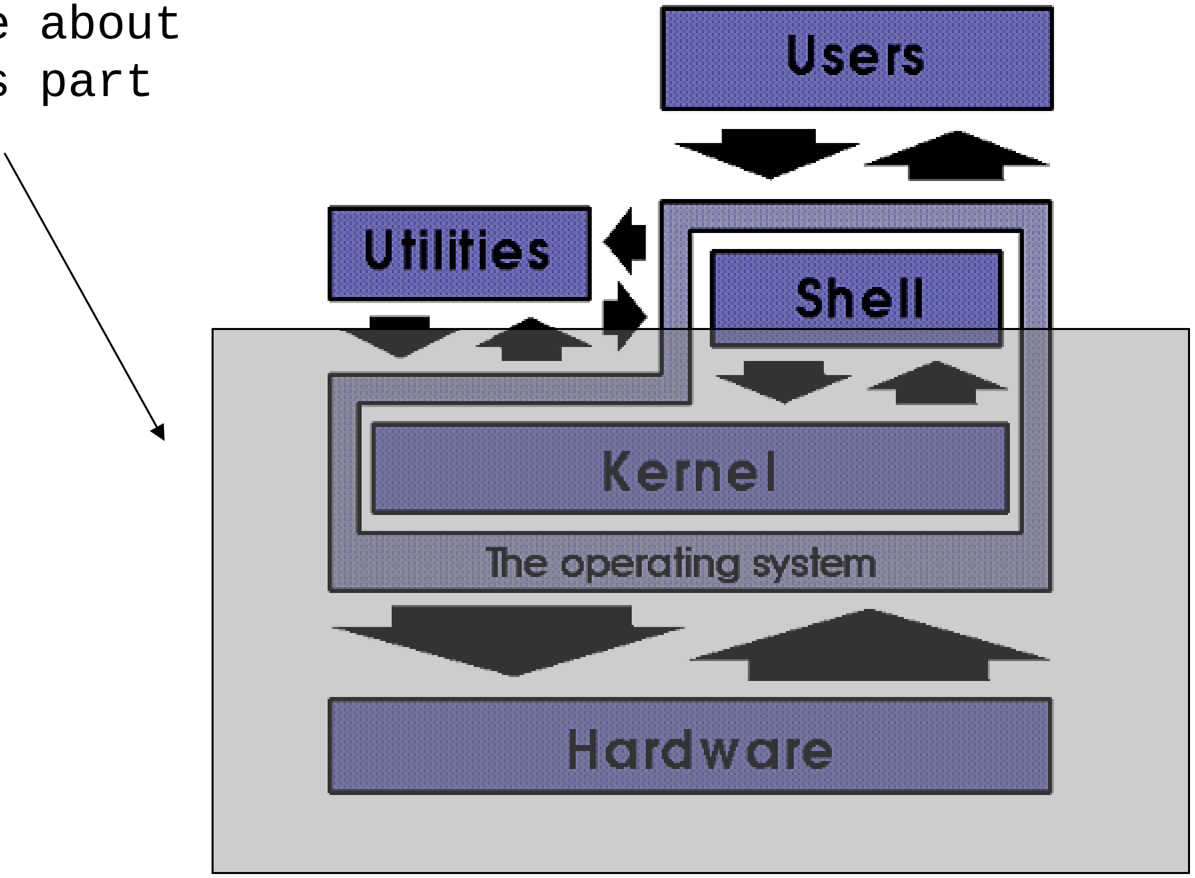
- Pass commands to UNIX by typing at the “command line”, also known as the "shell".
- Many bioinformatics programs have command line interfaces: BLAST, hmmer, EMBOSS, etc.
- Use commands to:
  - Move files around
  - Look at files
  - Search files
  - Much more
- STOP a command with Control-C

The Shell

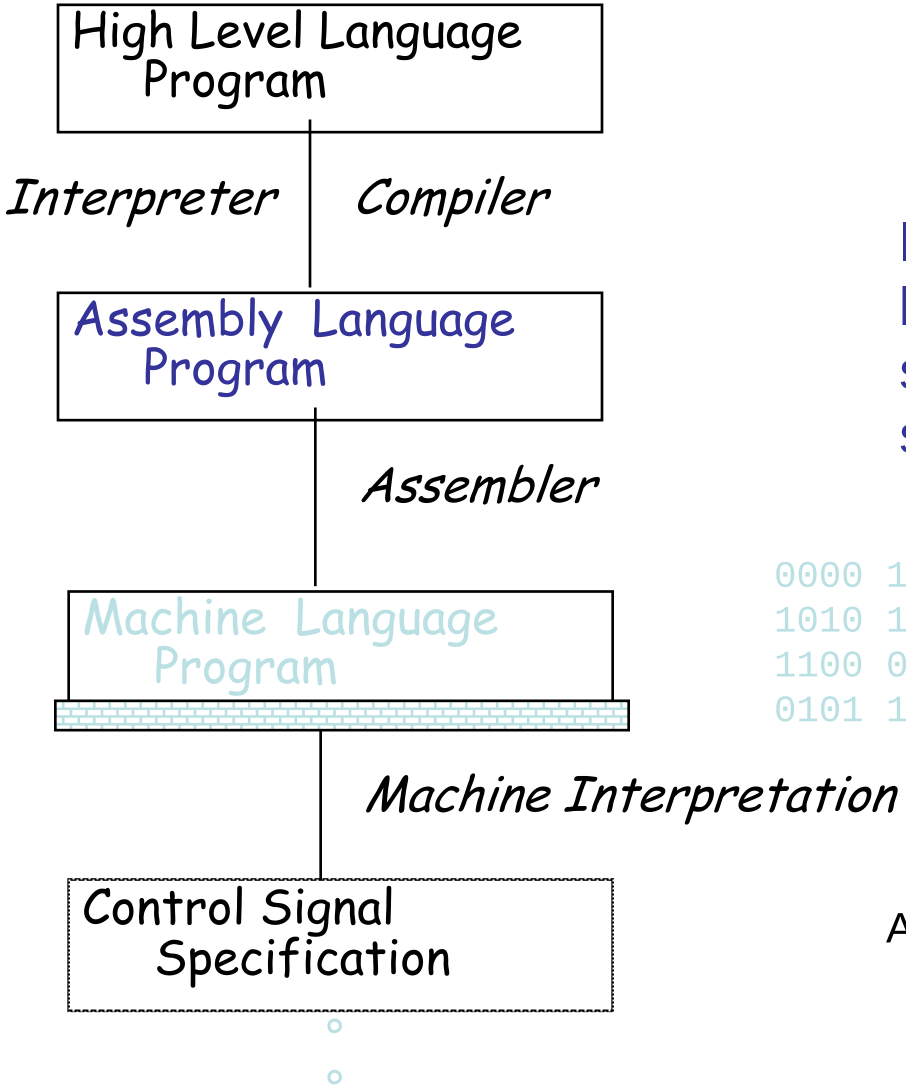
- User's interface with the rest of the system
  - Writes a prompt (like "`orchestra>`")
  - Waits for user input
  - Interprets user's (keyboard) commands
  - Executes one or more programs
  - Writes results and errors (or nothing at all) to the terminal window
  - Writes another prompt...
- The UNIX "butler"

Anatomy of the UNIX OS

You don't
care about
this part



Levels of Representation


High Level Language Program
Interpreter
Compiler
Assembly Language Program
Assembler
Machine Language Program
Machine Interpretation
Control Signal Specification

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw $15,   0($2)
lw $16,   4($2)
sw $16,   0($2)
sw $15,   4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

```
ALUOP[0:3] <= InstReg[9:11] & MASK
```

More on the Shell

- Shell commands can take flags and arguments.
- Shells can use wildcards ("globs") as arguments.
- Shells have a standard input (the keyboard) and output (the screen), which can be redirected.
- The shell is also a programming language that can handle variables, loops, etc.
- There are many different shells – sh, **bash**, csh, **tcsh**, ksh, zsh. They differ only in minor details.
- The shell is case sensitive

Anatomy of a Command

- Command [flags] [arguments]
  - Action [modifications] [object]
  - What to do [how to do it] [what to do it to]
- Command: what you want to do
  - The name of the program, like "clustalw"
  - Command must have a space after it
    (In general, separate things by spaces)

Anatomy of a Command

- Flags – “how to do it”
  - Usually start with - or --
  - Often just a dash and a letter or word: `ls -l`
  - May have arguments (`blastall -p blastp`)
- Arguments – “what to do it to”
  - Often one or more filenames or directories
  - Glob: `*.html` means all files ending with .html
  - Parameters: `grep 'some_text' myfile`

Anatomy of a Command

- Flags and arguments *may* be optional, depending on the command

– `ls`	list current directory contents
– `ls -l`	...in long format (which gives "permissions")
– `ls a b`	list files (or directories) a and b
– `ls -l a b`	...in long format

- Command example:

```
orchestra> blastall -p blastp -d nr -i
  in.fasta -o blast.out -e 1e-5 -v 10 -b 5
```

Playing with Files

- `cp` – copy
  - `cp file1 file2` creates file2
  - `cp file1 dir` - creates dir/file1
  - `cp file1 dir/file2` - creates dir/file2
- `mv` – move (rename - sort of like cp)
- `rm` – remove (delete)
- `touch` – change file timestamp

  OR create empty file

Warning: cp and mv will overwrite existing files!

- `man` – manual page (also try `info` on Linux)

Exercise 2 - Simple Commands

1. Create a file named `alice` with touch
2. Copy it to a new file named `bert`
3. Rename `bert` to `Alex` (Alex, not alex!)
4. See what files are in the directory
    1. How big are they?
    2. List just one file in long format
    3. Are there any "hidden" files? (`ls -a`)
    4. List all files starting with "a" (Do you get 1 or 2?)
5. Delete a file
6. See options to BLAST: type `blastall -`

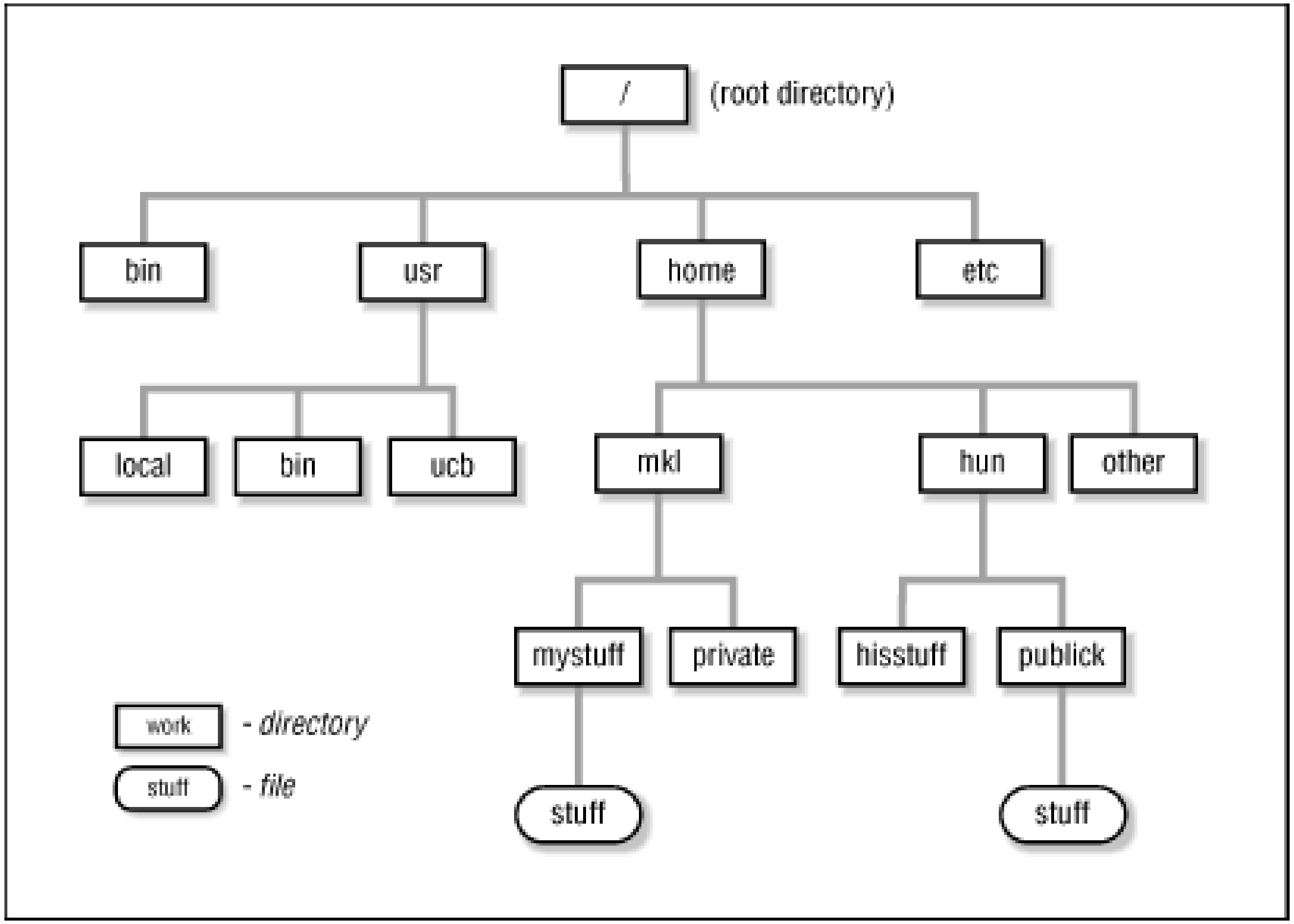
Outline

- Getting Started: What's Unix?
- Getting In: Logging into Unix
- Getting Stuff Done: Commands
- Getting Around: The Filesystem
- Getting Fancy: Complex Commands
- Getting CPU Time: Using the Cluster

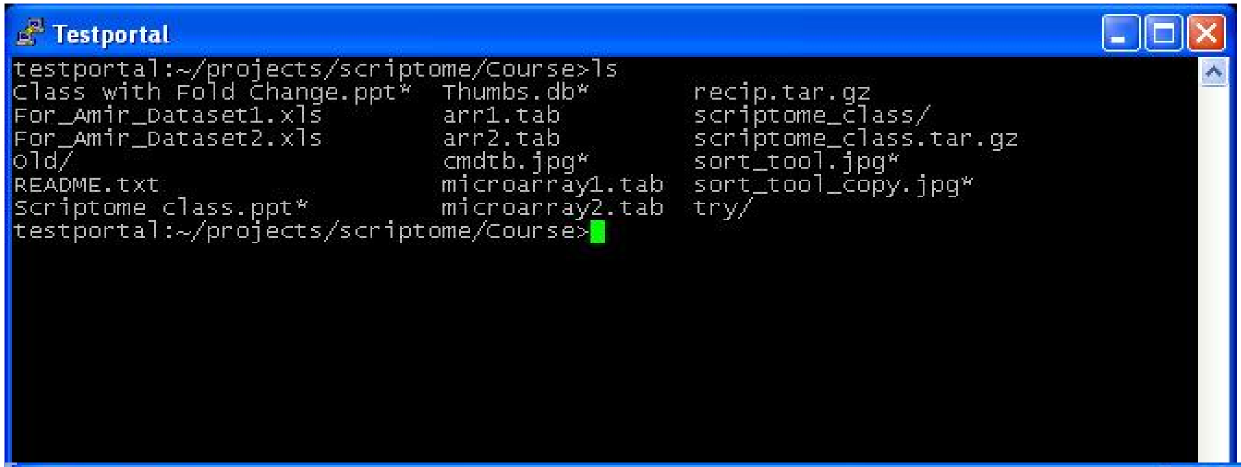
The UNIX Filesystem

- File system is a branching tree
- Folders contain:
  - Other folders, and/or
  - Files (text, Word, HTML, "binary", ...)
- Root directory (folder) is named “/”.
- Other directories have names like dir1, my_work, or Important_Data
- You are always "in" a "working directory"
- Join directories with "/" to show nesting:
  - `/home/lp28/sb302`

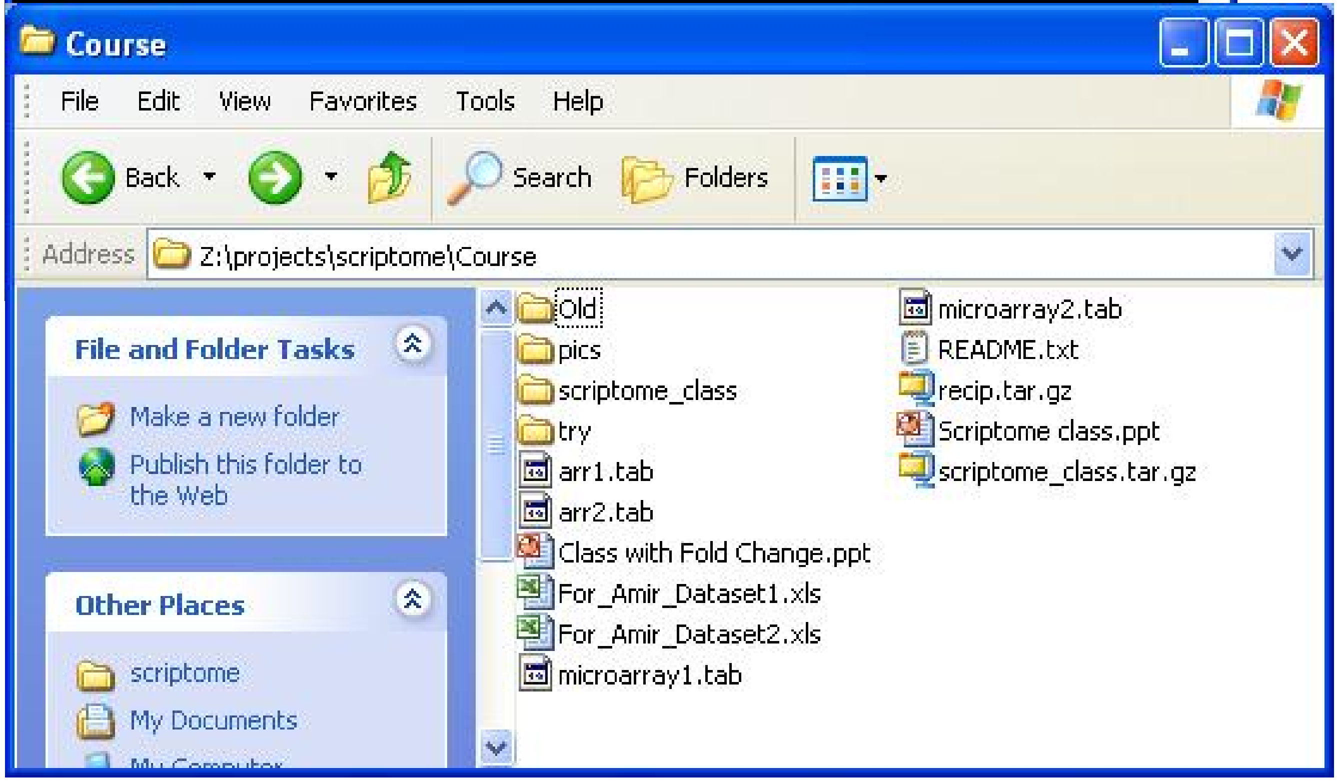
Filesystem Tree



Filesystems: UNIX and Windows



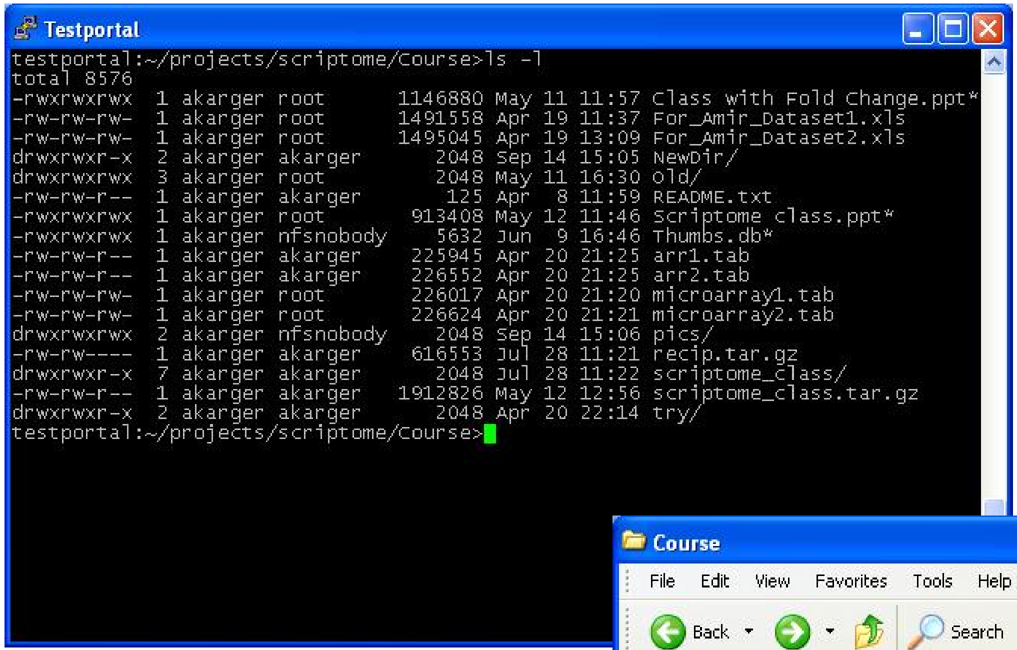
```
testportal:~/projects/scriptome/Course>ls
Class with Fold Change.ppt*  Thumbs.db*       recip.tar.gz
For_Amir_Dataset1.xls        arr1.tab         scriptome_class/
For_Amir_Dataset2.xls        arr2.tab         scriptome_class.tar.gz
old/                         cmdtb.jpg*       sort_tool.jpg*
README.txt                   microarray1.tab  sort_tool_copy.jpg*
Scriptome class.ppt*         microarray2.tab  try/
testportal:~/projects/scriptome/Course>
```



```
Z:\projects\scriptome\Course

Old                          microarray2.tab
pics                         README.txt
scriptome_class              recip.tar.gz
try                          Scriptome class.ppt
arr1.tab                     scriptome_class.tar.gz
arr2.tab
Class with Fold Change.ppt
For_Amir_Dataset1.xls
For_Amir_Dataset2.xls
microarray1.tab
```

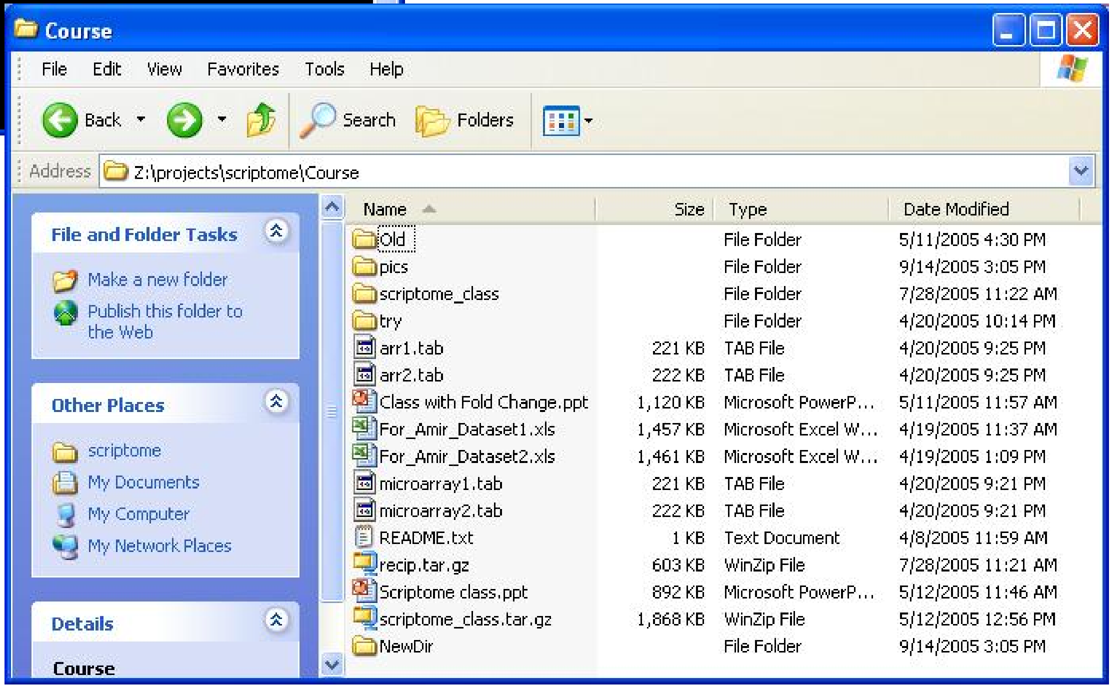
- Same info
- Different way of showing it

UNIX & Windows, cont.



```
testportal:~/projects/scriptome/Course>ls -l
total 8576
-rwxrwxrwx  1 akarger root      1146880 May 11 11:57 Class with Fold Change.ppt*
-rw-rw-rw-  1 akarger root      1491558 Apr 19 11:37 For_Amir_Dataset1.xls
-rw-rw-rw-  1 akarger root      1495045 Apr 19 13:09 For_Amir_Dataset2.xls
drwxrwxr-x  2 akarger akarger      2048 Sep 14 15:05 NewDir/
drwxrwxrwx  3 akarger root         2048 May 11 16:30 old/
-rw-rw-r--  1 akarger akarger       125 Apr  8 11:59 README.txt
-rwxrwxrwx  1 akarger root        913408 May 12 11:46 Scriptome class.ppt*
-rwxrwxrwx  1 akarger nfsnobody    5632 Jun  9 16:46 Thumbs.db*
-rw-rw-r--  1 akarger akarger    225945 Apr 20 21:25 arr1.tab
-rw-rw-r--  1 akarger akarger    226552 Apr 20 21:25 arr2.tab
-rw-rw-rw-  1 akarger root       226017 Apr 20 21:20 microarray1.tab
-rw-rw-rw-  1 akarger root       226624 Apr 20 21:21 microarray2.tab
drwxrwxrwx  2 akarger nfsnobody    2048 Sep 14 15:06 pics/
-rw-rw----  1 akarger akarger    616553 Jul 28 11:21 recip.tar.gz
drwxrwxr-x  7 akarger akarger      2048 Jul 28 11:22 scriptome_class/
-rw-rw-r--  1 akarger akarger   1912826 May 12 12:56 scriptome_class.tar.gz
drwxrwxr-x  2 akarger akarger      2048 Apr 20 22:14 try/
testportal:~/projects/scriptome/Course>
```

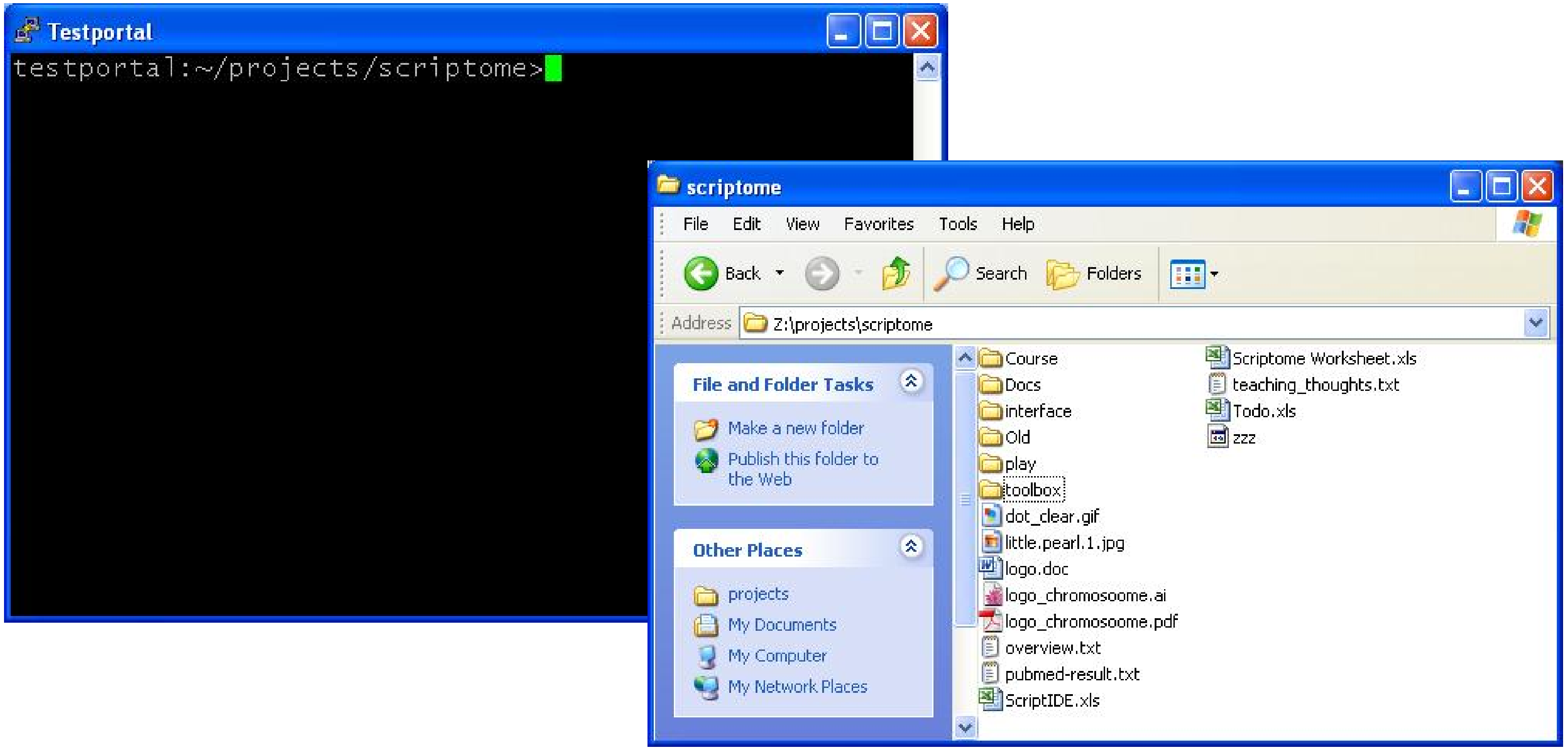
- "Details" view



Address: Z:\projects\scriptome\Course

Name	Size	Type	Date Modified
Old		File Folder	5/11/2005 4:30 PM
pics		File Folder	9/14/2005 3:05 PM
scriptome_class		File Folder	7/28/2005 11:22 AM
try		File Folder	4/20/2005 10:14 PM
arr1.tab	221 KB	TAB File	4/20/2005 9:25 PM
arr2.tab	222 KB	TAB File	4/20/2005 9:25 PM
Class with Fold Change.ppt	1,120 KB	Microsoft PowerP...	5/11/2005 11:57 AM
For_Amir_Dataset1.xls	1,457 KB	Microsoft Excel W...	4/19/2005 11:37 AM
For_Amir_Dataset2.xls	1,461 KB	Microsoft Excel W...	4/19/2005 1:09 PM
microarray1.tab	221 KB	TAB File	4/20/2005 9:21 PM
microarray2.tab	222 KB	TAB File	4/20/2005 9:21 PM
README.txt	1 KB	Text Document	4/8/2005 11:59 AM
recip.tar.gz	603 KB	WinZip File	7/28/2005 11:21 AM
Scriptome class.ppt	892 KB	Microsoft PowerP...	5/12/2005 11:46 AM
scriptome_class.tar.gz	1,868 KB	WinZip File	5/12/2005 12:56 PM
NewDir		File Folder	9/14/2005 3:05 PM

UNIX vs. Win: Changing Directories



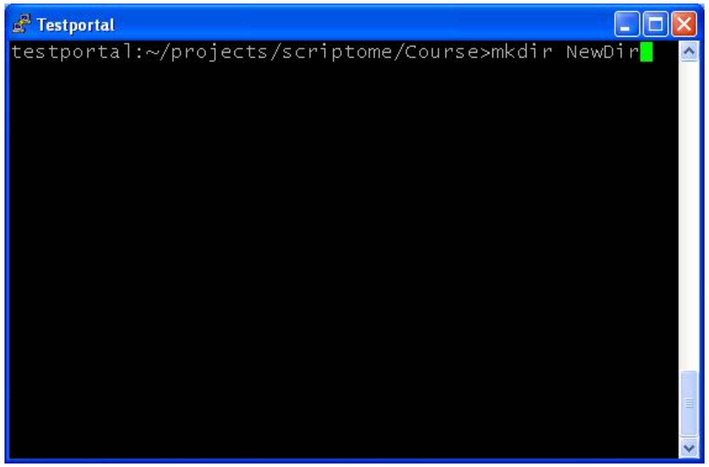
```
testportal:~/projects/scriptome>
```

Address: Z:\projects\scriptome

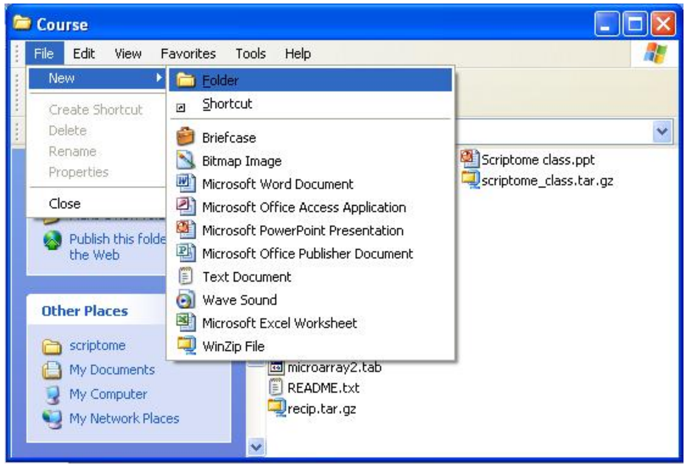
```
Course                 Scriptome Worksheet.xls
Docs                   teaching_thoughts.txt
interface              Todo.xls
Old                    zzz
play
toolbox
dot_clear.gif
little.pearl.1.jpg
logo.doc
logo_chromosoome.ai
logo_chromosoome.pdf
overview.txt
pubmed-result.txt
ScriptIDE.xls
```

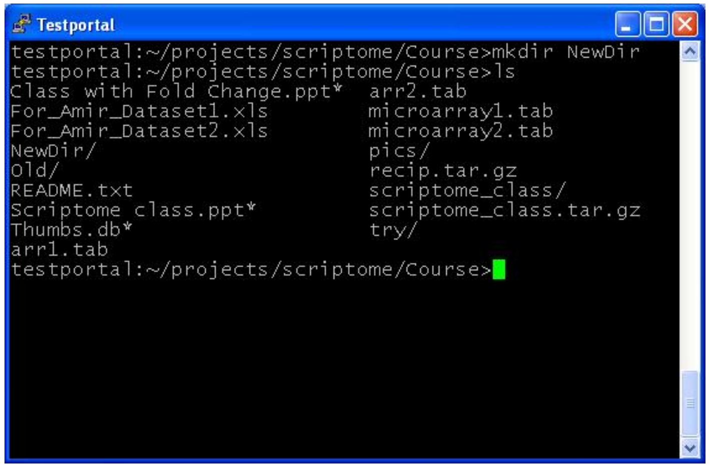
- Same operations
- Different ways of showing it

UNIX vs. Win: Make a directory

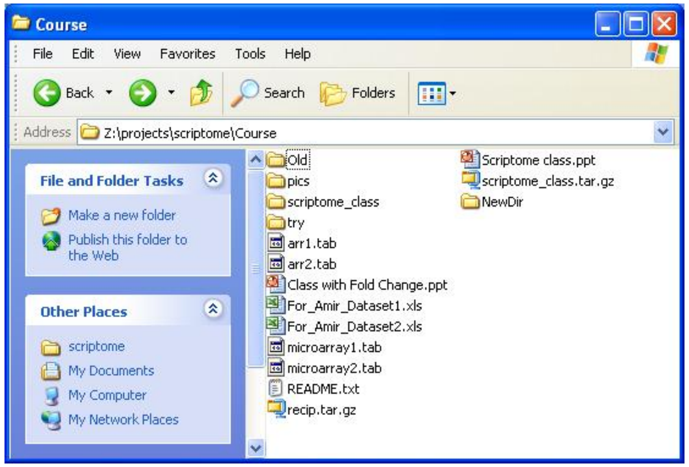


```
testportal:~/projects/scriptome/Course>mkdir NewDir
```





```
testportal:~/projects/scriptome/Course>mkdir NewDir
testportal:~/projects/scriptome/Course>ls
Class with Fold Change.ppt*  arr2.tab
For_Amir_Dataset1.xls        microarray1.tab
For_Amir_Dataset2.xls        microarray2.tab
NewDir/                      pics/
Old/                         recip.tar.gz
README.txt                   scriptome_class/
Scriptome class.ppt*         scriptome_class.tar.gz
Thumbs.db*                   try/
arr1.tab
testportal:~/projects/scriptome/Course>
```



Home Directory

- Your personal file space
- When you login, this is your working directory
- Read, write, and delete files here
  - Or in any directory inside this directory
  - Not true everywhere
- You can delete everything!
  - But it won't break anybody else's stuff or the overall system
- On orchestra, user fred’s home directory is `/home/fred`
  - I.e., directory "fred" inside "home" inside "/"

Getting Around the Filesystem

- pwd - where am I?
- cd – change directory
- ls - list directory(ies) and/or files
- mkdir - make directory
- rmdir - remove a directory

~ - home directory

. - current directory

.. - one directory up from .

Many Ways to Refer to a File

- If user `fred` logs in and does `cd mydir`, all of the following refer to the file `myfile` in that directory:
  - `myfile`
  - `./myfile`
  - `/home/fred/mydir/myfile` ("Full path" starts with /)
  - `~/mydir/myfile`
  - `../mydir/myfile (or ../../home/fred/myfile)`
- Referring to file `myfile2` in sub-directory `dir2`:
  - `dir2/myfile2`
  - `/home/fred/mydir/dir2/myfile2`
  - `~/mydir/dir2/myfile2`

Naming Files II

- Up one directory and down into a different one:
  - `../other_dir/file3`
  - `/n/home/fred/other_dir/file3`
  - `~/other_dir/file3`
- Use any of these in a command:
  - `mv myfile ~/upfile ../other_dir/file3 dir4`
- Commands can also have paths
  - `ls` actually does `/bin/ls`

    (Unix magically looks in the right place for built-in commands)
  - `../my_program –I ../some_dir/myfile`

Exercise 3

1. What directory are you in?
2. Copy the file `unixclass.tar.gz` from `/usr/tmp` to your home directory
3. Copy a file into `/usr/tmp` (but don't overwrite anything!)
4. Delete the file from `/usr/tmp`
5. Make a new directory `mydir`
    1. Move two files into `mydir`
    2. Make copies of them (with new names) in `mydir`
6. List all the files in `mydir` AND your home directory using one command
7. Now move into `mydir` and list them again

Working with files

- **more** - Scroll through a file page by page. (Works even with files that are too big to be opened by a text editor.)
- **head** - View top 10 lines of a file. head -n 3 views 3 lines
- **tail** - View the tail (bottom) of a file. tail -f to view a growing file.
- **wc** - Count words, lines and characters in a file
- **grep** - Filter a file for lines matching (or *not* matching) a pattern
- **gzip (gunzip)** - Compress (uncompress) a file.
- **tar** - Archive a whole directory into one file (or unarchive)

Many programs (wc, grep, etc.) can work on multiple files

grep

- Grep searches for lines of text that match a specific pattern

```
% grep 'gene' myfile
```

  - Prints any line containing “gene” in file myfile
  - Also "genetic" or "Eugene"
  - NOT "Gene" or "gENe" (use grep -i for that)
  - Putting quotes around the search string let you look for spaces or special characters
- `grep -v 'gene' myfile`
  - prints lines NOT containing "gene"

Exercise 4

1. Uncompress and unarchive `unixclass.tar.gz`
   1. `gunzip unixclass.tar.gz`
   2. `tar –xvf unixclass.tar`
2. Move into `unixclass`
3. Fun with sequence files
   1. Read the first sequence from moreseqs
   2. Read the last sequence from moreseqs
   3. Get all the FASTA IDs from moreseqs (hint: *what do all ID lines have in common?)*
   4. Is the sequence CLERH in the file? ABCDE?
   5. How many lines are in moreseqs? What about seqs and moreseqs together?

Outline

- Getting Started: What's Unix?
- Getting In: Logging into Unix
- Getting Stuff Done: Commands
- Getting Around: The Filesystem
- Getting Fancy: Complex Commands
- Getting CPU Time: Using the Cluster

Command line editing

Until you press <Enter>, you can go back over the command line and edit it using the keyboard.

- Backspace - Delete the previous character and back up one.
- Left arrow, right arrow - Move the text insertion point (cursor) one character to the left or right.
- TAB does command/filename completion
  - Type `ls mores` and then TAB
  - UNIX finishes the filename `moreseqs`

Command history

- UNIX stores a history of your commands
- Up arrow, down arrow - Move up and down in the command history.
  - Modify a command if desired
  - Hit <Enter> to redo that command
- `history 10` - lists last 10 commands with numbers
- !135 will rerun command 135 from the history list

Redirecting output

- What if your command creates lots of output? What if you want to store the output?
- The ">" character **redirects** your output into a file, instead of to the screen

```
% grep 'Hsp' a.fasta b.fasta > blah.Hsp
```

  - Get all lines from 2 files with "Hsp" in them
  - Warning: this will get the description lines for genes with Hsp OR "this gene is not at all similar to Hsp"
- Warning: ">" will overwrite any existing file!

Redirecting output II - Appending

- Use ">>" to append

```
% grep 'Lys' a.fasta b.fasta >> blah.Hsp
```

- Add Lys genes to the list from before
- Now we can read, edit, play with our results
- OR, don't use an intermediate file at all...

The Pipe “|”

- The unix pipe “|” is used to chain together multiple commands.
- The **output** of one command is used as the **input** for the next command

```
% ls –la | more
```

  – Pass the possibly long output of ls to a more which will let you view the output one page at a time

```
% ls –ltr | tail -n 1
```

  – Sort files by modification date ascending, view only the most recently modified file

Exercise 5

- Run the EMBOSS program "transeq" on a FASTA sequence file (/opt/emboss/bin/)
  - just type transeq and it will ask you for input
- BLAST the translated sequence:
  - blastp (protein-protein blast).
  - Blast against the `Homo_sapiens.aa` database
  - Create output file `opsd_human.blp.`
  - Blast syntax looks like:

```
blastall –p blastp –i myseq –d my_database [-o my_out]
```

- Get hits with "rhodopsin" in their name
- Count the hits with "rhodopsin" in their name (Hint: Use a pipe, and count lines)

Exercise 5

- Run the emboss program "transeq"

```
/opt/emboss/bin/transeq opsd_human.fasta
```

- Run BLAST on your sequence

```
blastall –p blastp –i opsd_human.pep –d
/rodeo/databases/blast/Homo_sapiens.aa -o
opsd_human.blp
```

- Get hits with "rhodopsin" in their name

```
grep 'rhodopsin' opsd_human.blp
```

- Count hits with "rhodopsin" in their name

```
grep 'rhodopsin' opsd_human.blp | wc
```

File/Directory Naming Practices

- Use letters, numbers, period and underscore in filenames
- Use lower-case letters. The file Alpha.txt is different from alpha.txt. You'll never remember whether the filename has a capital letter or not.
- Use common file extensions. E.g., save a text file as blah.txt. (Required in Windows, NOT in UNIX)
- Filenames starting with a dot (.) are hidden files.
- Make names short, but not cryptic. Use correctly-spelled nouns when possible. Store inventory in inventory.dat and not inv.dat.
- Don't use spaces. (For a Windows file with spaces in it, use quotes)

```
    cp 'My Windows File.doc' blah.doc
```

- Avoid naming a file with the same name as a Unix command. You can find out if a name is a Unix command by using the man command.

Outline

- Getting Acquainted: What's Unix?
- Getting In: Logging into Unix
- Getting Stuff Done: Commands
- Getting Around: The Filesystem
- Getting Fancy: Complex Commands
- Getting CPU Time: Using the Cluster

The Orchestra Cluster

- The cluster has over 160 computers
- The computers are shared among HMS and other Harvard biology researchers.
- But some researchers are greedy
- Who decides how to share resources?
- LSF - **Load Sharing Facility**

LSF

- Users submit their jobs (e.g., BLAST)
- Jobs go into queues
- LSF selects which job to run next based on:
  - Current load conditions
  - Resources requirements of the applications
  - How important you are
- With LSF, remotely run jobs behave just like jobs run on the local host. (Even graphical jobs!)
- With LSF, computer resources are shared fairly, without wasting idle computers

Queues

- Queues have different maximum run times
  - Your job will be killed if it exceeds that time
- Queues have different priorities
- Some queues can only be used by certain users
- You choose which queue to submit your job to
  - short - 1 hour, high priority
  - normal - 24 hrs, normal priority. **Default queue**
  - long - unlimited, low priority
  - interact - run graphical programs from the cluster
  - *shared_int_2h, shared_2h, all_2h, all_1d*
  - *sysbio_2h, cbi_unlimited* - it's a secret

LSF Commands

- See https://wiki.med.harvard.edu/Orchestra/IntroductionToLSF
- Or just do "man bsub" on portal

bsub

bjobs

bkill

bhist

bqueues

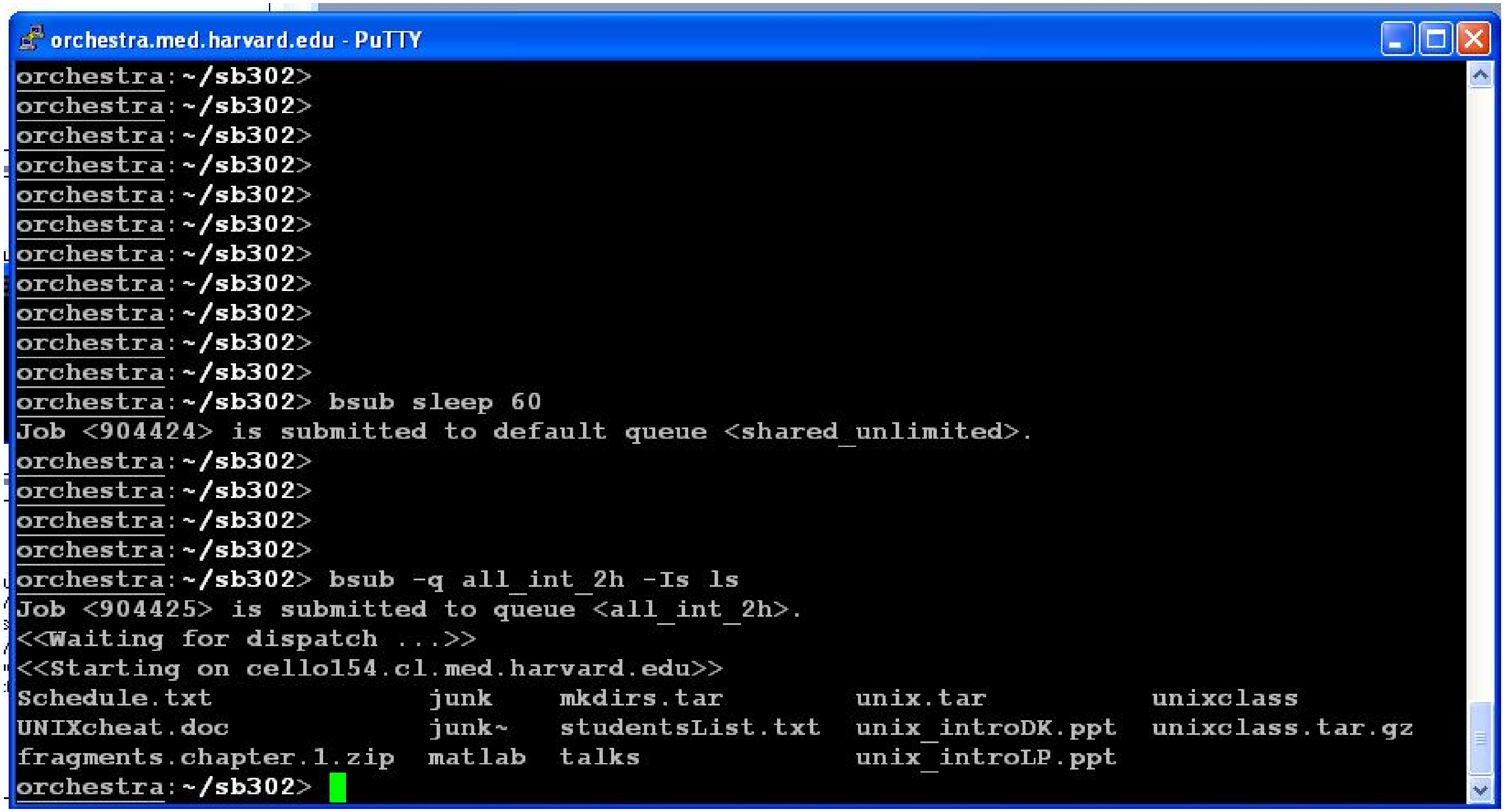
bpeek

lsid

lsload

bhosts

bsub


orchestra.med.harvard.edu - PuTTY

```
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302> bsub sleep 60
Job <904424> is submitted to default queue <shared_unlimited>.
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302> bsub -q all_int_2h -Is ls
Job <904425> is submitted to queue <all_int_2h>.
<<Waiting for dispatch ...>>
<<Starting on cello154.cl.med.harvard.edu>>
Schedule.txt               junk     mkdirs.tar        unix.tar          unixclass
UNIXcheat.doc              junk~    studentsList.txt  unix_introDK.ppt  unixclass.tar.gz
fragments.chapter.1.zip    matlab   talks             unix_introLP.ppt
orchestra:~/sb302>
```

- Most important LSF command. Submit your job(s) to the LSF system.

bsub

- Just put "bsub" before the command and arguments you would type anyway
- This submits your job to the LSF system.
- Jobs usually wait in the queue for seconds to minutes before starting, depending on:
  - How busy the queue is
  - How many other jobs you're running/have run recently
  - Memory or other requirements for your job

bsub flags

- **bsub [bsub flags] command [command flags] [command arguments]**
- Remember to put bsub flags **before** the command!
- bsub flags: there are many, but most are unnecessary
- `bsub -q all_1d blastall -p blastp -i in.fasta -d nr`

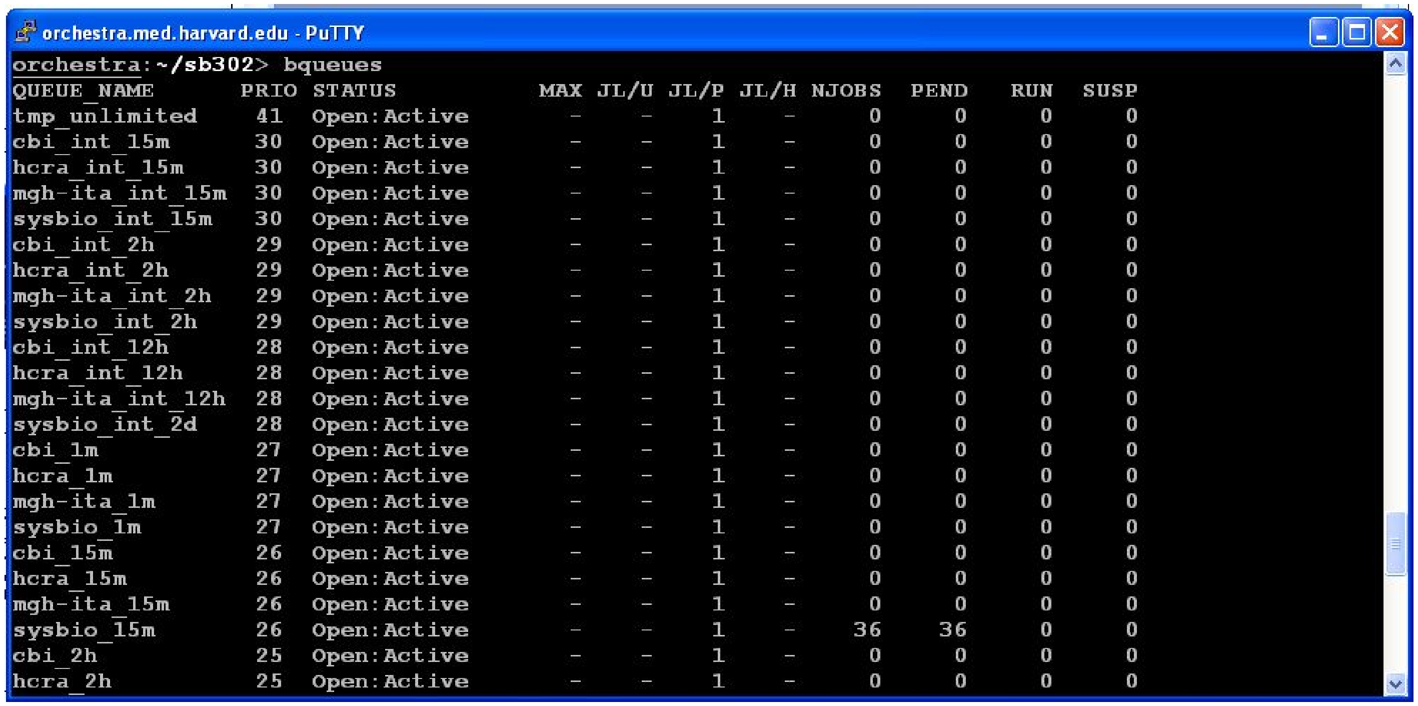
  send blast job to the all_1d queue
- `bsub -q all_12h -m violin059 blastall ...`

  send blast job to the all_12h queue, run on host violin050
- `bsub -R "rusage[mem=1000]" -o jobout myscript`

  send myscript to default queue, request 1G of memory, and send the job's stdout and stderr to the file called "jobout".
  - Note: if -o is not used, and the program tries to write to the screen, you will, by default, receive the entire job output via email.

bqueues

bqueues: lists the queues in the system. You may not be allowed to use all of the queues



orchestra.med.harvard.edu - PuTTY

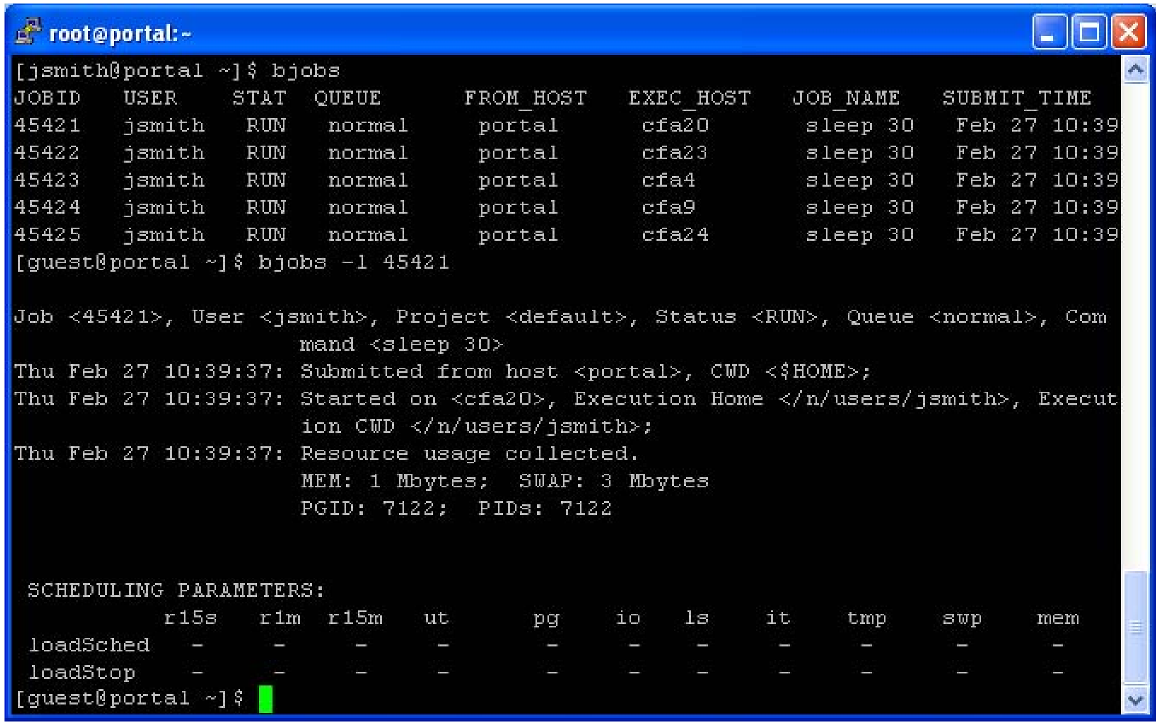
```
orchestra:~/sb302> bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
tmp_unlimited    41  Open:Active       -    -    1    -     0     0     0     0
cbi_int_15m      30  Open:Active       -    -    1    -     0     0     0     0
hcra_int_15m     30  Open:Active       -    -    1    -     0     0     0     0
mgh-ita_int_15m  30  Open:Active       -    -    1    -     0     0     0     0
sysbio_int_15m   30  Open:Active       -    -    1    -     0     0     0     0
cbi_int_2h       29  Open:Active       -    -    1    -     0     0     0     0
hcra_int_2h      29  Open:Active       -    -    1    -     0     0     0     0
mgh-ita_int_2h   29  Open:Active       -    -    1    -     0     0     0     0
sysbio_int_2h    29  Open:Active       -    -    1    -     0     0     0     0
cbi_int_12h      28  Open:Active       -    -    1    -     0     0     0     0
hcra_int_12h     28  Open:Active       -    -    1    -     0     0     0     0
mgh-ita_int_12h  28  Open:Active       -    -    1    -     0     0     0     0
sysbio_int_2d    28  Open:Active       -    -    1    -     0     0     0     0
cbi_1m           27  Open:Active       -    -    1    -     0     0     0     0
hcra_1m          27  Open:Active       -    -    1    -     0     0     0     0
mgh-ita_1m       27  Open:Active       -    -    1    -     0     0     0     0
sysbio_1m        27  Open:Active       -    -    1    -     0     0     0     0
cbi_15m          26  Open:Active       -    -    1    -     0     0     0     0
hcra_15m         26  Open:Active       -    -    1    -     0     0     0     0
mgh-ita_15m      26  Open:Active       -    -    1    -     0     0     0     0
sysbio_15m       26  Open:Active       -    -    1    -    36    36     0     0
cbi_2h           25  Open:Active       -    -    1    -     0     0     0     0
hcra_2h          25  Open:Active       -    -    1    -     0     0     0     0
```

- `Bqueues` all queues
- `bqueues -u lp28` only queues you can submit to
- `bqueues -l long` info about the "long" queue

bjobs

bjobs: lists your jobs currently active in the system. This includes jobs that are pending (waiting to be dispatched for execution) and those executing.

- `bjobs -uall`
  list of all jobs by all users
- `bjobs -l 45322`
  details on a particular job



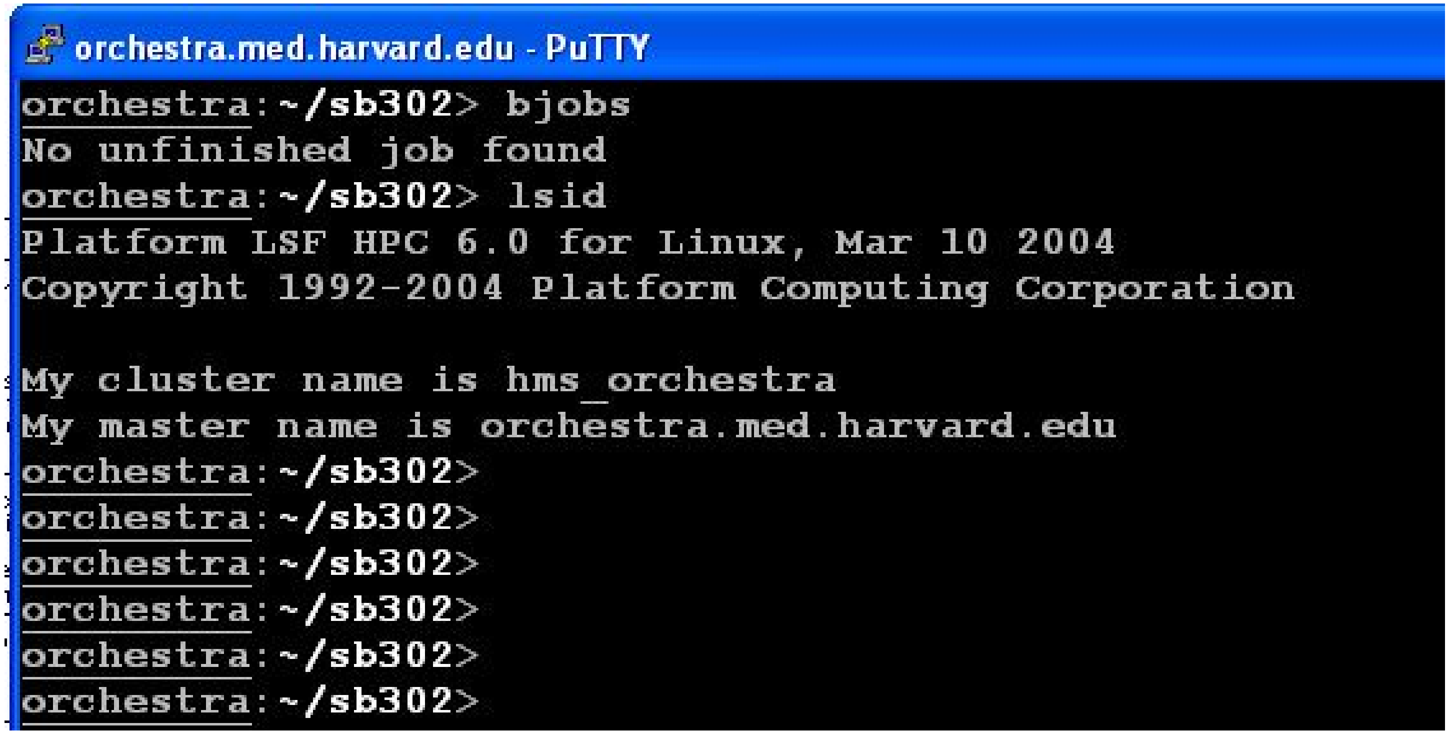
```
[jsmith@portal ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
45421   jsmith   RUN   normal     portal      cfa20       sleep 30   Feb 27 10:39
45422   jsmith   RUN   normal     portal      cfa23       sleep 30   Feb 27 10:39
45423   jsmith   RUN   normal     portal      cfa4        sleep 30   Feb 27 10:39
45424   jsmith   RUN   normal     portal      cfa9        sleep 30   Feb 27 10:39
45425   jsmith   RUN   normal     portal      cfa24       sleep 30   Feb 27 10:39
[guest@portal ~]$ bjobs -l 45421

Job <45421>, User <jsmith>, Project <default>, Status <RUN>, Queue <normal>, Com
                     mand <sleep 30>
Thu Feb 27 10:39:37: Submitted from host <portal>, CWD <$HOME>;
Thu Feb 27 10:39:37: Started on <cfa20>, Execution Home </n/users/jsmith>, Execut
                     ion CWD </n/users/jsmith>;
Thu Feb 27 10:39:37: Resource usage collected.
                     MEM: 1 Mbytes;  SWAP: 3 Mbytes
                     PGID: 7122;  PIDs: 7122


 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -
[guest@portal ~]$
```

Other commands

- **bhist:** Returns a list of your jobs that have finished more than 1 hour ago. For jobs that have finished within the last hour, use the bjobs command.
- **bpeek:** Returns the stdout (standard output) of your job. Bascially you can "peek" at the output** of your job while its running, to see whats happening.
  **Unless you specified an output file.
- **bkill:** This command kills your job(s). You can specify one or more jobs to kill. You can kill all your jobs if you specify a zero, like "bkill 0".

lsid

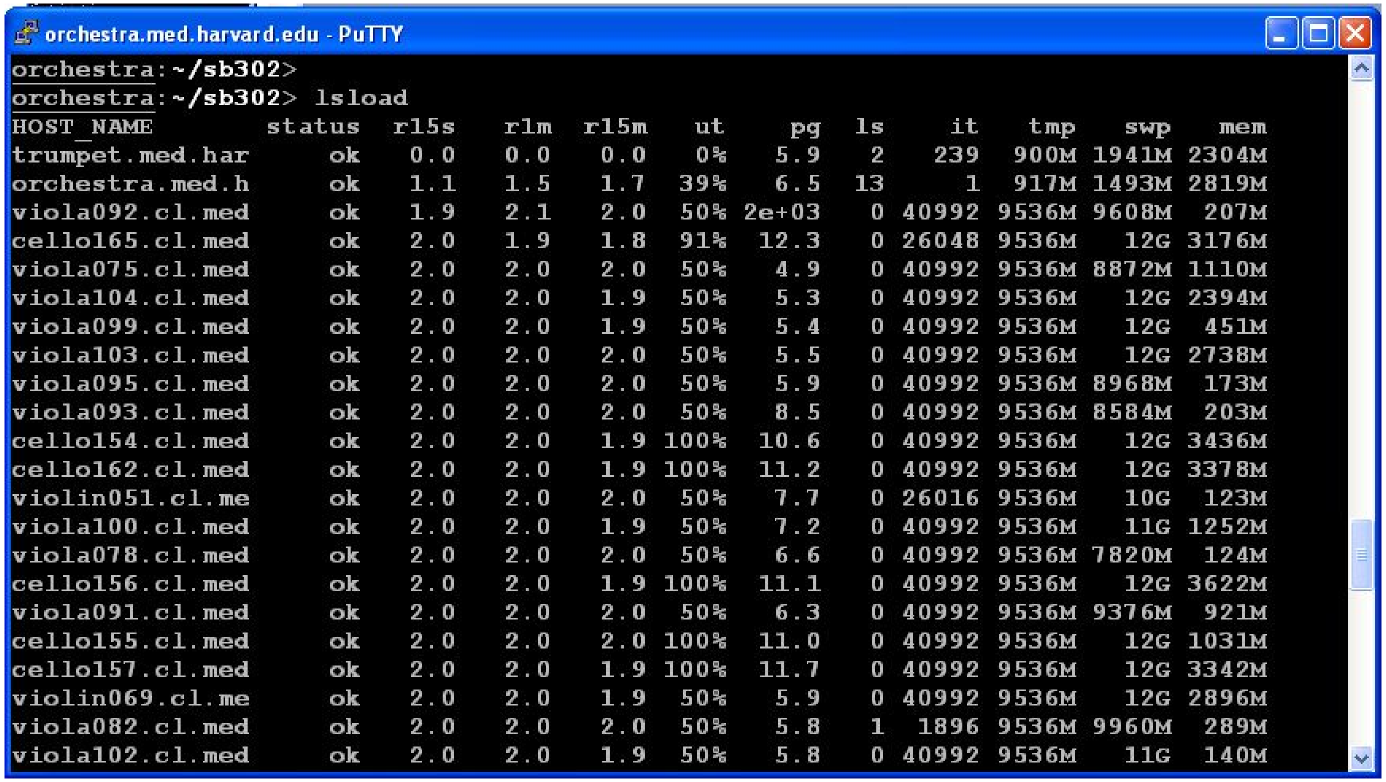
- **lsid** returns the cluster name and the master host.
- This command verifies that you are connected to the LSF cluster.



```
orchestra:~/sb302> bjobs
No unfinished job found
orchestra:~/sb302> lsid
Platform LSF HPC 6.0 for Linux, Mar 10 2004
Copyright 1992-2004 Platform Computing Corporation

My cluster name is hms_orchestra
My master name is orchestra.med.harvard.edu
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
orchestra:~/sb302>
```

lsload

- **lsload** returns the current load of the hosts in the cluster. It lists the various hosts and how busy they are.

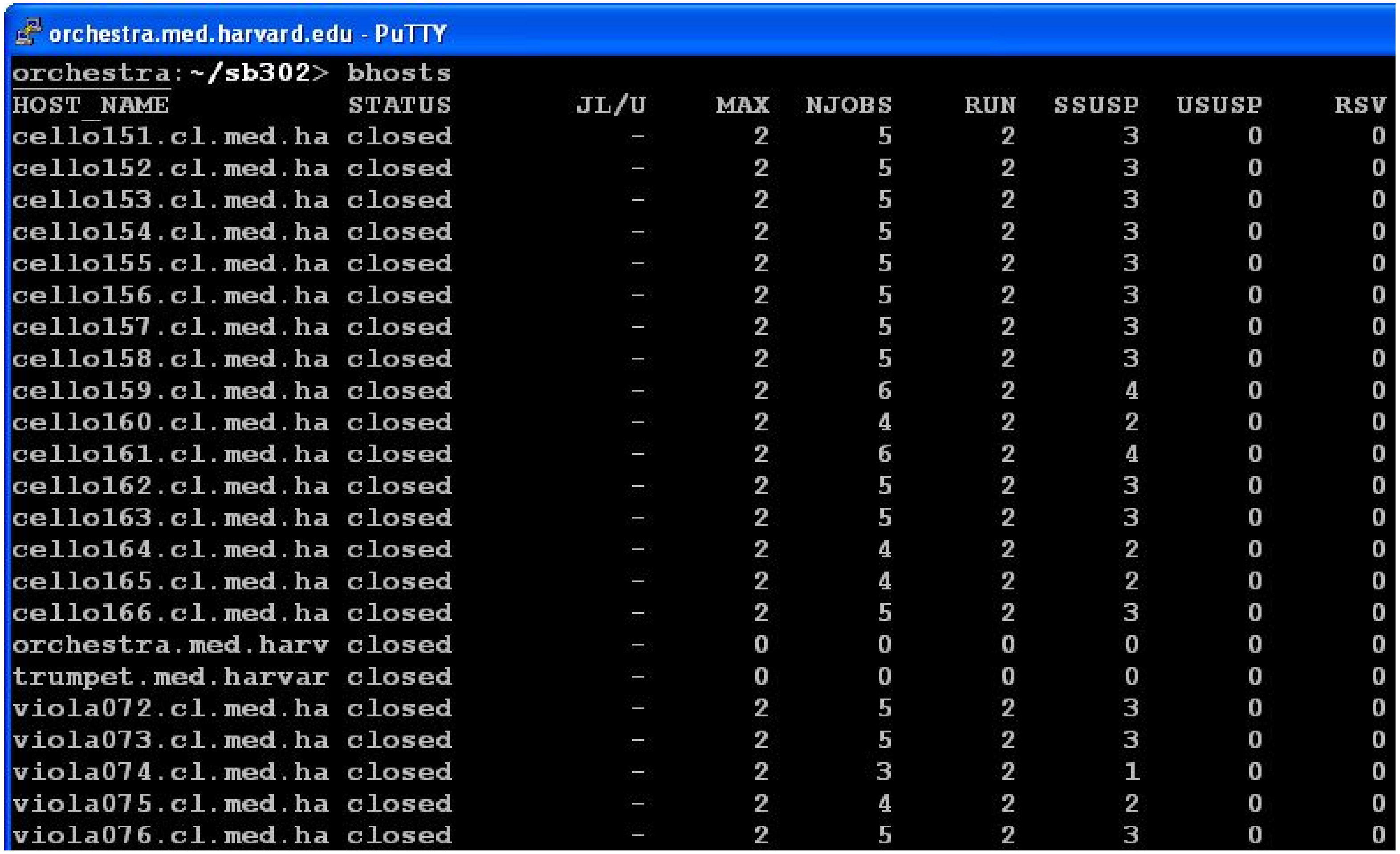


orchestra.med.harvard.edu - PuTTY

```
orchestra:~/sb302>
orchestra:~/sb302> lsload
HOST_NAME       status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem
trumpet.med.har     ok   0.0   0.0   0.0   0%   5.9   2   239  900M 1941M 2304M
orchestra.med.h     ok   1.1   1.5   1.7  39%   6.5  13     1  917M 1493M 2819M
viola092.cl.med     ok   1.9   2.1   2.0  50% 2e+03   0 40992 9536M 9608M  207M
cello165.cl.med     ok   2.0   1.9   1.8  91%  12.3   0 26048 9536M   12G 3176M
viola075.cl.med     ok   2.0   2.0   2.0  50%   4.9   0 40992 9536M 8872M 1110M
viola104.cl.med     ok   2.0   2.0   1.9  50%   5.3   0 40992 9536M   12G 2394M
viola099.cl.med     ok   2.0   2.0   1.9  50%   5.4   0 40992 9536M   12G  451M
viola103.cl.med     ok   2.0   2.0   2.0  50%   5.5   0 40992 9536M   12G 2738M
viola095.cl.med     ok   2.0   2.0   2.0  50%   5.9   0 40992 9536M 8968M  173M
viola093.cl.med     ok   2.0   2.0   2.0  50%   8.5   0 40992 9536M 8584M  203M
cello154.cl.med     ok   2.0   2.0   1.9 100%  10.6   0 40992 9536M   12G 3436M
cello162.cl.med     ok   2.0   2.0   1.9 100%  11.2   0 40992 9536M   12G 3378M
violin051.cl.me     ok   2.0   2.0   2.0  50%   7.7   0 26016 9536M   10G  123M
viola100.cl.med     ok   2.0   2.0   1.9  50%   7.2   0 40992 9536M   11G 1252M
viola078.cl.med     ok   2.0   2.0   2.0  50%   6.6   0 40992 9536M 7820M  124M
cello156.cl.med     ok   2.0   2.0   1.9 100%  11.1   0 40992 9536M   12G 3622M
viola091.cl.med     ok   2.0   2.0   2.0  50%   6.3   0 40992 9536M 9376M  921M
cello155.cl.med     ok   2.0   2.0   2.0 100%  11.0   0 40992 9536M   12G 1031M
cello157.cl.med     ok   2.0   2.0   1.9 100%  11.7   0 40992 9536M   12G 3342M
violin069.cl.me     ok   2.0   2.0   1.9  50%   5.9   0 40992 9536M   12G 2896M
viola082.cl.med     ok   2.0   2.0   2.0  50%   5.8   1  1896 9536M 9960M  289M
viola102.cl.med     ok   2.0   2.0   1.9  50%   5.8   0 40992 9536M   11G  140M
```

bhosts

bhosts: Returns the list of hosts (computers) that are part of the LSF system.



orchestra.med.harvard.edu - PuTTY

```
orchestra:~/sb302> bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
cello151.cl.med.ha closed          -      2      5      2      3      0      0
cello152.cl.med.ha closed          -      2      5      2      3      0      0
cello153.cl.med.ha closed          -      2      5      2      3      0      0
cello154.cl.med.ha closed          -      2      5      2      3      0      0
cello155.cl.med.ha closed          -      2      5      2      3      0      0
cello156.cl.med.ha closed          -      2      5      2      3      0      0
cello157.cl.med.ha closed          -      2      5      2      3      0      0
cello158.cl.med.ha closed          -      2      5      2      3      0      0
cello159.cl.med.ha closed          -      2      6      2      4      0      0
cello160.cl.med.ha closed          -      2      4      2      2      0      0
cello161.cl.med.ha closed          -      2      6      2      4      0      0
cello162.cl.med.ha closed          -      2      5      2      3      0      0
cello163.cl.med.ha closed          -      2      5      2      3      0      0
cello164.cl.med.ha closed          -      2      4      2      2      0      0
cello165.cl.med.ha closed          -      2      4      2      2      0      0
cello166.cl.med.ha closed          -      2      5      2      3      0      0
orchestra.med.harv closed          -      0      0      0      0      0      0
trumpet.med.harvar closed          -      0      0      0      0      0      0
viola072.cl.med.ha closed          -      2      5      2      3      0      0
viola073.cl.med.ha closed          -      2      5      2      3      0      0
viola074.cl.med.ha closed          -      2      3      2      1      0      0
viola075.cl.med.ha closed          -      2      4      2      2      0      0
viola076.cl.med.ha closed          -      2      5      2      3      0      0
```

Running an LSF job

```
# Run a BLAST job on the cluster
bsub –q all_2h blastall –p blastp –i seqs.pep –d nr –o
   b.out

#!/bin/tcsh
# This "shell script" runs many BLAST jobs in parallel.
# It uses one node (one job) for each input FASTA file.
# If you save it as multiblast.sh then you can run it like
#    tcsh multiblast.sh
foreach file ( *fasta )
   bsub –q all_2h blastall –p blastp –i $file –o $file.blast.out
end
```

Advanced UNIX Topics

“UNIX - that undiscovered country, from whose Bourne shell no executable returns...”

“Abandon hope, all ye who hit Enter here”

Standard Output and Error

```
some_long_program > long.out
```

- If there’s an error, I don’t want to wait for the whole program to finish to find out
- So (well-behaved) programs split output:
  - standard output (`stdout`) has regular info
  - standard error (`stderr`) has errors and warnings
- Both go to the screen by default
- > and >> only redirect `stdout` to a file
- >& and >>& will redirect `stdout` AND `stderr`
- `bsub -o` redirects `stdout` and `stderr` to a file

Command line editing

- Control-A (^A): Move cursor to beginning of line. Mnemonic: A is first letter of alphabet
- ^E: **E**nd of line

(^Z was already taken for something else).

- ^D: **D**elete character currently under the cursor.
- ^K: **K**ill (cut) from the cursor to end of line. (Deleted text goes to a clipboard)
- ^Y: **Y**ank (paste) the clipboard text back onto the command line

UNIX Scripting

- UNIX shell has a whole programming language
  - Variables, loops, conditions, etc.
  - Language is slightly different for `bash` vs. `tcsh`
  - Examples are `tcsh` unless otherwise noted

```
portal> foreach i (*seqs)
foreach? echo $i
foreach? grep –c ‘WAR’ $i
foreach? end
```

UNIX Scripting II

- Create scripts using text editors:
  - `pico` (good for beginners), `vi` (Vim), `emacs`
- Run scripts by
  - `chmod +x blah.sh`
  - `./blah.sh`
  - Or just `tcsh blah.sh`
- Commands, loops, etc. run as if you typed them in at the command line

```
foreach i (*seqs)
  echo $i
  grep –c ‘WAR’ $i
end
```

UNIX Scripting III

- `./myscript a b c`
  - $1 is “a”, $2 is “b”, $3 is “c”
  - print, compare, etc. the $ variables in script
  - The set command creates normal variables
- Conditions:

```
if ($1 == 1) then
  echo “hi”
else
  echo “bye”
endif
```

- Read tcsh (or bash) man pages for much more

Login rc Files

- Some scripts automatically run when you login
  - `tcsh: /etc/csh.cshrc, /etc/csh.login, .tcshrc`
  - `bash: /etc/profile, /etc/bashrc, .bashrc`
- These are just regular shell scripts
- Put commands in here that you want to run every time you login

The UNIX Prompt

```
[botka@portal ~ 1 ]%
```

- 1st field – user
- 2nd field – hostname
- 3rd field –directory
- 4th field – command number
- Prompt character

- You can customize your prompt (man tcsh)

```
if ($?prompt) then
  set prompt='[%n@%m %c %h ]% ‘
endif
```

Environment Variables

- Information about your account
- Preferences for your account
- Locations of databases, files, programs
- `tcsh:`
  - `setenv BLASTDB ~/my_blastdbs`
  - `printenv BLASTDB`
- `bash:`
  - `BLASTDB = ~/my_blastdbs`
  - `echo $BLASTDB`

The UNIX $PATH

- `PATH` is an environment variable set up by the system
- Lists the places where the shell looks for executable files (`ls` is really `/bin/ls`)
- Set automatically, but you can add to it
- Change it in your `.tcshrc/.bashrc` file.
  - `bash: set path=( ~/bin $path )`
  - `tcsh: setenv PATH “~/bin $PATH”`

File/Directory Permissions

- Every file and directory has an owner (a user) and a group
- `groups akarger` – groups I belong to
- `ls -l` says each file’s owner/group
- `chown`, `chgrp` changes these values

```
botka@portal.77% ls -la
total 64
drwxrwxr-x    2 botka    botka         4096 Feb 25 08:06 ./
drwxrwx---   48 botka    cgradmin     45056 Feb 25 08:02 ../
-rw-rw-r--    1 botka    botka         5332 Feb 25 08:02 moreseqs
-rw-rw-r--    1 botka    botka         1102 Feb 25 08:06 opsd_human.fasta
-rw-rw-r--    1 botka    botka         1247 Feb 25 08:02 seqs
```

Permissions II

- `ls -l` says who can do what to a file/directory:
  - r: read, w: write (or delete), x: execute a file, see inside a directory
  - categories: **u**ser, **g**roup, **o**ther
- `chmod` changes these values
  - `chmod o+w seqs` (now others can edit the file)
  - `chmod 644 seqs` (magic to set permissions: see chmod man page)

```
botka@portal.77% ls -la
total 64
drwxrwxr-x    2 botka    botka        4096 Feb 25 08:06 ./
drwxrwx---   48 botka    cgradmin    45056 Feb 25 08:02 ../
-rw-rw-r--    1 botka    botka        5332 Feb 25 08:02 moreseqs
-rw-rw-r--    1 botka    botka        1102 Feb 25 08:06 opsd_human.fasta
-rw-rw-r--    1 botka    botka        1247 Feb 25 08:02 seqs
```

More Shortcuts: Aliases and Links

- `ln -s ../../some/far/away/file ./here`
  – ln is just like cp, but it makes a link instead
  – more here will more the far away file, etc.
- `alias cdd ‘cd some/far/away/dir’`
  – put this in your .tcshrc so you always have it
- alias can also use variables!
  – `alias lastlog ‘set lastlog=`ls –dtr /usr/local/adm/log/updatedb/{\!:*}* | tail –n 1`; echo “Most recent \!:* log: $lastlog”; more $lastlog`

More commands

- /bin has 81 commands
  - And then there’s /usr/bin, /usr/local/bin...
- Data manipulation: sort, cut, paste, join, tr
- File filters: sed, awk
- Real programming languages: Perl, Python