Shell Usage Outline

-What is the shell?

- An interface for running programs and interacting with the OS and File System.
- 5 common shells: borne (sh), bourne-again (bash), korn (ksh), c-shell (csh), and tsch
- default on Clear is tsch, default on sugar is bash (differences are mostly mini-language, such as setenv vs export for setting environment variable)

-Environment variables

- they are simply variables that hold strings
- these variables are inherited by programs run in the shell
- to access the value of the environment variable in the shell, prepend the name with a $
- the PATH variable tells the shell where to look for commands typed into the shell, ':' delimit different directories. We modify it so the hjc and hj commands can be found.
- JAVA_HOME is needed so that the java compiler and jvm know where the needed libraries are located
- HJ_HOME is set for the same reason but for the HJ compiler and subsystem
- each shell inherits several environment variables on start, use the printenv command to see them

-Paths

- absolute paths begin with '/' (called root, akin to C:)
- relative paths don't have a starting '/' and are resolved relative to your current directory
- '.' is a special directory name for the current directory
- '..' is a special directory name for the parent directory
- spaces need to be escaped by backslash or the whole path wrapped in quotes, otherwise the path is interpreted as separate arguments delimited by the spaces
- tab will autocomplete directory or file names up to the first conflict

-Common Shell Commands

1. **cd** <dir> - change directory, takes a path
2. **mv** <src> <dest> - moves a file from src to dest, -R will move a folder
3. **cp** <src> <dest> - like move, but copies
4. **ps** - see current processes running, suspended, or terminated (but not yet harvested)
5. **rm** <fileOrDirName> - remove file, -R for directories. BE CAREFUL, there is no recovering what you delete. If you use the -f flag, rm will not ask confirmation about deleting each file. rm -Rf / will wipe the computer (hopefully you don't have permissions to do that)
6. **mkdir** <dirname> - creates a directory of the name given
7. **chmod** [args] <filename>- changes permissions
8. **touch** <filename> - changes modification and access times if the file already exists, creates a new file otherwise
9. **ls** - list contents of current directory
10. **pwd** - display path to working directory (current directory)
11. **echo** <string> - prints the string to the screen.
12. **cat** <filename> - prints the contents of file to the screen
13. **grep** <string> <filename> - prints the lines in file that contain the string
14. **printenv** - prints out all the environment variables and their values
15. **man** <cmdname> will display info regarding the specified command

-.Xrc Files

- a shell script that is executed before the first shell prompt is presented to you
- each shell has a .<shellname>rc file for itself, which is located in your home folder
- ideal place to do setups (like setting up HJ_HOME, JAVA_HOME, and PATH) so they are always ready on start

-vim Text Editor

- usage vim <filename> to open a file, without filename it reads from stdin
- when you open a file with vim, you open it in command mode, which means your keys do something OTHER THAN TYPING THEIR VALUES, so don't do anything until you read the next bit.
- the arrow keys move the cursor around
- the keys h,j,k,l also move the cursor around
- dd will delete the current line
- a will change to append mode, and let you type after the cursor, press escape to exit append mode
- i will change to insert mode, and let you type at the cursor, escape to exit
- in command mode, you can type : for a few other commands and press enter to execute them
- w will write your changes to your file without exiting vim
- w <filename> will write to the specified file name overwriting it if it exists, or creating it if it doesn't.
- q will quit if no changes haven't been saved.
- q! will quit and discard unsaved changed.
- typically to exit you'll use either :wq to save and quit, or :q! to discard your changes.
- pressing escape will exit you out of colon commands
- There are A TON of other commands to look up

-SSH and SFTP

- ssh stands for Secure Shell
- basically used for remote login
- login format is %>ssh <username>@<address>

- typical usage is <netid>@crystal.clear.rice.edu
- you'll prompted for password then, just use your netID password (except for Sugar)
- sftp stands for SSH File Transfer Protocol
- typical commands are same as a shell, but are executed on the remote machine
- prepend commands with an 'l' to execute on the local machine (ex. cd changes directory of remote machine, lcd changes local directory)
- you cannot execute programs in sftp
- %>put <src file> <dest file> copies the file from local to remote, renaming it if you specify a name different than the original
- %>get <src file> <dest file> is put's inverse, copies from remote to local
- the mput and mget variants will do their respective functions for multiple files.
- Typically you'll specify a regular expression (well a subset anyway) instead of the file name.
- %>mget * will get all files in the current directory of the remote machine.
- %>mput *.hj will put all files ending in .hj in the local machine's current directory to the remote machine's current directory
- quit, bye, or exit will log you out of ssh and sftp

-Signals

- ctrl-C will interrupt the current running program, and get you back to the shell
- ctrl-D sends EOF

-IO Redirection

- <cmd> **<** <filename> operator redirects stdin from the command line to file
- <cmd> **>** <filename> operator redirects stdout from the command line to file
- <cmd1> **|** <cmd2> 'pipe' operator redirects output of cmd1 as input to cmd2
- the operators can be combined