CS 235 - Useful UNIX/Linux commands for CS 235

beginning notes:

- **remember:** UNIX/Linux is **case-sensitive**!
- `nrs-projects.humboldt.edu` is the host name of a (virtual) HSU computer that has Java installed on it, along with the Oracle middleware to allow JDBC to connect to the HSU Oracle database named `student`.
- you will use `ssh` to **connect** to `nrs-projects` to try out JDBC.
- once you have logged onto `nrs-projects.humboldt.edu`, here is a collection of UNIX/Linux commands that you might find useful:

help-related commands:

`man` *desired_command*	display the UNIX/Linux manual page for *desired_command*, if it exists
`apropos` *string*	display names of UNIX/Linux commands followed by 1-line descriptions for commands whose 1-line descriptions contain *string*

directory-related commands:

`cd`	change directory; make the home directory the current working directory
`cd` *directory_name*	change the current working directory to *directory_name*
`.`	a nickname for the current directory
`..`	a nickname for the parent of the current directory
`~`*username*	a nickname for *username*'s current directory
`~`	a nickname for the current user's home directory
`pwd`	give the name of the current (present) working directory
`mkdir` *directory_name*	make a new directory named *directory_name* within/under the current working directory
`rmdir` *directory_name*	remove the directory *directory_name* within/under the current working directory; note that it must be **empty** for this to work
`ls`	list the contents of the current working directory
`ls -l`	...in "long" format, including file permissions
`ls -ld`	...including permissions and information for subdirectories instead of their contents
`ls` *directory_name*	list the contents of the directory *directory_name*
`chmod 700` *directory_name*	protect the directory *directory_name* so that only **you** can read, write, or execute its contents. **This should be used for homework directories.**

file-related commands:

`cp` *filename newfilename*	create a copy of *filename* with the name *newfilename*
`cp` *f1 f2 f3 ... directory_name*	creates copies of files *f1*, *f2*, *f3*, ... (all that you care to list) in the directory *directory_name*
`mv` *filename newfilename*	change the name of the file *filename* to *newfilename*
`mv` *f1 f2 f3 ... directory_name*	moves files *f1*, *f2*, *f3*, ... (all that you care to list) to the directory *directory_name*
`rm` *filename*	remove the file *filename* **(be careful - this cannot be undone!)**
`rm -i` *filename*	slightly-safer way to remove a file -- asks you to confirm removal! (**BUT still cannot be undone!**)
`chmod 600` *filename*	protect the file *filename* - only **you** can read or write it
`more` *filename*	look at the contents of *filename* on-screen, one screen at a time
`cat` *filename*	look at the contents of *filename* on-screen, all at once
`nano` *filename* `vi` *filename* `emacs` *filename*	edit file *filename* (these are three different **text editors** available on `nrs-projects`)

commands and tips for stopping a UNIX/Linux process:

^C	(typing ctrl key and letter c at the same time) This can often be used to stop or kill a running UNIX/Linux command (a command running in the foreground). Useful if you accidentally type a command that does more than you want to see (e.g., when you don't want to see the rest of a man page)
`ps x`	gives information about currently-running processes that you own (even from other UNIX/Linux sessions). The name of each process is on the far right, and the **process id** of each process is in the first column. (**Beware**: the options for `ps` vary on different flavors of UNIX/Linux!)
`kill` *process_id* `kill -9` *process_id*	stop, or kill, the process with process id *process_id*. I was always taught to try the version **without** -9 **before** trying the version with -9, because the former kills the process less "messily".

other commands and etc.:

`javac` *ClassName*`.java`	compile the Java source code in the file *ClassName*`.java` into Java bytecode; results in at least the Java bytecode file *ClassName*`.class`, and possibly additional Java bytecode files (one per non-`public` class also contained within *ClassName*`.java`'s source code)

`java` *ApplicClassName*	use the Java Virtual Machine (JVM) to interpret and run the Java application class `ApplicClassName` (whose Java bytecode is in *ApplicClassName*`.class`); starts execution at its `main` method
`javadoc` *ClassName*`.java`	generate HTML documentation for *ClassName* based on its javadoc-style comments
`*`	UNIX/Linux wildcard character that matches **any** 0 or more characters. E.g., `ha*s` matches `has`, `ha3s`, `happiness`, etc.
`?`	UNIX/Linux wildcard character that matches **any single** character. E.g., `ha?s` matches `hams`, `ha3s` but does **not** match `has`, `haaas`
tab key	in several UNIX/Linux shells (including `nrs-projects`' default shell, `bash`), typing this key after you have started typing a file name will cause the shell to try to **complete** (fill in) the file name you have started typing, if it can. This is called **filename completion**.
`grep` *pattern* `*`	look for files in the current working directory that **contain inside** of them the pattern or letters *pattern*
`diff` *file1 file2*	compare the contents of *file1* and *file2*, and show any differences. If the two files are identical, nothing is returned.
`history`	show a list of the most recently-executed commands in this UNIX/Linux session
!!	redo the last UNIX/Linux command executed
!`com`	redo the most recent UNIX/Linux command executed starting with the letters `com`
!−*num*	redo the UNIX/Linux command executed *num* commands ago
!*num*	redo the UNIX/Linux command numbered *num* in the history list
up-arrow key	lets you scroll through the commands in the history list
`quota`	On many UNIX/Linux/Linux systems, this lets you know how much of you disk space quota you are using. This does not seem to be set up on `nrs-projects` at this time, but just in case it is set up at some point, I'm still including it in this list.