Linux Command Summary

Directory Navigation and Listing	
`cd`	change to home directory
`cd ..`	go up to parent directory
`cd subdir`	change to subdirectory `subdir`
`ls`	list content of current directory
`ls -l`	list content with details
`ls -a`	list content including hidden files

File Commands	
`cp src dest`	copy `src` file to `dest` file
`cp -r sDir dDir`	copy "recursively" `sDir` directory to `dDir` directory (copies subdirectories too)
`mv src dest`	move - renames `src` as `dest`
`rm fileName`	removes file `fileName`
`rm -r dirName`	removes directory recursively
`rmdir dirName`	removes empty `dirName`
`mkdir dirName`	makes directory called `dirName`
`chmod 750 file1`	change permission of `file1` by specifying a three digit octal # where digits are owner, group, world each octal digit in binary are: read (4) ,write (2) ,execute (1)
`cat file1`	display `file1` to screen
`less file1`	display `file1` with pagination (space - next page, q-exit, ↑,↓- keys)

Process Management	
`ps`	List processes with pid
`top`	Shows the real-time processes
`kill -9 pid`	Kills the process with `pid #`

Keyboard Shortcuts	
`<tab>`	Auto-complete partial file name
`<Ctrl>+c`	Kill current command/program
`<Ctrl>+z`	Sleep current program
`<↑>`	Recall previous command(s)
`<Ctrl>+d` `exit`	log-off and close terminal

"Programming" Tools	
`nano file.cpp`	Simple text-editor
`emacs file.cpp`	Better C/C++ editor

`gcc file.cpp`
`g++ file.cpp`
`-o exeFile`
	C compiler: compile to `a.out`
	C++ compiler: compile to `a.out`
	Options:
	compile to `exeFile` instead

`./a.out`	execute program in current directory (".") called `a.out`
`time exeFile`	run `exeFile` and print timing when done
`script out.txt`	capture output to file `out.txt` `<Ctrl>+d` to end

1) Log-on to student.cs.uni.edu using a Telnet/ssh client (e.g., PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/)
(On a MAC you can probably use: `ssh userName@student.cs.uni.edu` in a terminal to log-on)

2) Your initial log-in is the same as your UNI CatID with initial password of: `1234`

3) For this activity I want you to:

- create and then move into a directory called `lecture4` to store today's files
- use an editor (emacs or nano) to write a simple C++ program that prompts the user for their age, allows them to enter it, and outputs it back for them. Use the file name `age.cpp`
- compile the C++ to an executable file called `age` using: `g++ -o age age.cpp`
- when its working capture the interactive running of the program using: `script out.txt` to start the capture and `<Ctrl>+d` to end the capture
- display the contents of the `out.txt` to the screen using the `less` command

4) Use a secure ftp client (e.g., FileZilla: https://filezilla-project.org) to copy `lecture4` to local computer
(On a MAC you can probably use: `scp -r localDir userName@student.cs.uni.edu:/lecture4`)

5) On your local computer zip the `lecture4` directory and submit as Homework #1 at:
http://www.cs.uni.edu/~fienup/cs1160f13/homework/index.htm

Operator Precedence and Associativity		
Operator	**Associativity**	**Usage(s)**
`::`	unary: left-to-right binary: right-to-left	
`() [] -> .`	left-to-right	parenthesis index object pointer/structure pointer dot operator
`++ -- - + ! ~ (type) * & sizeof`	right-to-left	increment and decrement unary negation and plus logical negation one's complement operator type cast indirection address-of/reference
`* / %`	left-to-right	multiply, division, remainder
`+ -`	left-to-right	addition and subtraction
`<< >>`	left-to-right	io: insertion and extraction bit-wise shift left and right
`< <= > >=`	left-to-right	comparisons for inequality
`== !=`	left-to-right	comparison for equality
`&`	left-to-right	bit-wise AND
`^`	left-to-right	bit-wise exclusive-OR
`|`	left-to-right	bit-wise OR
`&&`	left-to-right	logical AND
`||`	left-to-right	logical OR
`?:`	right-to-left	conditional
`= += -= *= /= %= &= ^= |= <<= >>=`	right-to-left	assignment
`,`	left-to-right	comma operator

Header File	Function Example	Description
cmath	`y = abs(x)`	Returns the absolute value of an integer argument

	`y = cos(x)`
	`y = sin(x)`
	`y = tan(x)`
		Returns the cosine (sin or tangent) of the argument expressed in radians. Both the argument and returned value are doubles.

	`y = fmod(x, z)`	Returns the remainder of x divided by z where both arguments and the returned value are doubles
	`y = log(x)`	Returns the natural logarithm of the argument. The argument and the return type are doubles.
	`y = log10(x)`	Returns the base-10 logarithm of the argument. The argument and the return type are doubles.
	`y = sqrt(x)`	Returns the square root of the argument. The argument and the return type are doubles.
	`y = pow(x, z)`	Returns the first argument raised to the second argument power.
cstdlib	`y = rand()`	Returns the a pseudorandom number as an int.
	`srand(seed)`	The unsigned int argument acts as a seed value to the pseudorandom number generator.
ctime	`seed = time(0)`	Returns the number of seconds elapsed since midnight, Jan. 1, 1970. (The result can be used as the seed to srand)