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 -aux | grep uname`	List processes for `uname`
`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.c`	Simple text-editor
`emacs file.c`	Better C/C++ editor (see emacs handout)

`gcc file.c`
`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: `1234temp`

3) For this activity I want you to:

- create and then move into a directory called `hw4` to store files for this assignment
- use an editor (emacs or nano) to write a simple C program that prompts the user for their name and age, allows them to enter it, and outputs it back for them (on next back page). Use the file name `age.c`
- compile the C++ to an executable file called `age` using: `gcc -o age age.c`
- when its working, capture the interactive running of the program using: `script out.txt` to start the capture, `./age` to run the program, and `<Ctrl>+d` to end the capture
- display the contents of the `out.txt` to the screen using the `less out.txt` command (q-to exit `less`)

4) Use a secure ftp client (e.g., FileZilla: https://filezilla-project.org) to copy `hw4` to local computer
(On a MAC you can probably use: `scp -r userName@student.cs.uni.edu:hw4 localDir`)

5) On your local computer, zip the `hw4` directory and submit as Homework #4 at:
http://www.cs.uni.edu/~fienup/cs2420f14/homework/submissionDirections.htm

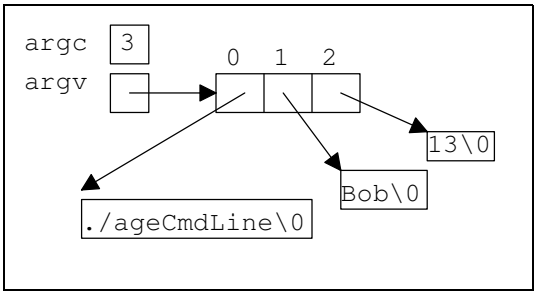
```c
/* File: age.c
   Compile by: gcc age.c
   Run by:  ./a.out
*/

#include <stdlib.h>
#include <stdio.h>

const int SIZE = 100;

int main(int argc, char * argv[]) {
  char name[SIZE];
  int age;

  printf("Enter your name: ");
  scanf("%s",name);

  printf("Enter your age: ");
  scanf("%d", &age);

  printf("%s your age is %d.\n", name, age);

  return 0;

} // end main
```

NOTES:

1) array names contain pointers to the first element of array
2) C only has pass-by-value, so we must explicitly pass the address of (using '&') a variable to a function changing it

```c
#include <stdlib.h>
#include <stdio.h>

// function prototypes

void getName(char []);
void getAge(int *);

const int SIZE = 100;

int main(int argc, char * argv[]) {
  char name[SIZE];
  int age;

  getName(name);

  getAge(&age);

  printf("%s your age is %d.\n", name, age);

  return 0;

} // end main

void getName(char name[]) {
  printf("Enter your name: ");
  scanf("%s",name);
} // end getName

void getAge(int * age) {
  printf("Enter your age: ");
  scanf("%d", age);
} // end getAge
```

```c
/* File:  ageCndLine.c
   Compile by: gcc -o ageCmdLine ageCmdLine.c
   Run by: ./ageCmdLine Bob 13
   Output:  Bob your age is 13.
 */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

const int SIZE = 100;

int main(int argc, char * argv[]) {
  int age;
  char * name;

  if (argc != 3) {
    printf("Usage %s firstName #age\n", argv[0]);
    exit(-1);
  }

  name = (char *) malloc(sizeof(char)*(strlen(argv[1])+1));
  strcpy(name, argv[1]);

  sscanf(argv[2],"%d",&age);

  printf("%s your age is %d.\n", name, age);

  return 0;

} // end main
```



NOTES:

3) argc is an integer containing the number of command-line arguments including the executable file name
4) argv is an array of char-pointers to the the start of each null-terminated string for the command-line arguments