Notes

2.1 Structs in C

* A struct in the C programming language is a data type that defines a grouped set of variables that will be referred to by one name. Structs are stored in a single, contiguous block of memory, with a fixed size: struct declarations may only consist of fixed size variables.
* Example of a basic Struct + definitions
* In order to create space to hold a string, we need to use malloc() or calloc() to create the space first. Then, we can use strcpy() to copy the characters into the space created by the dynamic memory allocation functions.
* Pointers to Structs
  + Here is how we could store a pointer to a struct from another:
    - struct room startRoom;
    - struct room midRoom1;
    - struct room midRoom2;
    - startRoom.numOutboundConnections = 2;
    - startRoom.outboundConnections[0] = &midRoom1;
    - startRoom.outboundConnections[1] = &midRoom2;
  + In order to refer to a member element of a struct through an address that we happen to have of that struct, we use the arrow operator "->" instead of the dot operator ".":
    - printf("Name: %s\n", startRoom.outboundConnections[0]->name);
* Copying Structs
  + Since struct variables are so large, it is usually best to pass around a pointer to a struct, rather than pass it by value.
  + If you do need to copy a struct, you use the basic assignment operator "=" to do so. This operator copies the value of each member variable in the struct on the right side of the assignment operator across to each matching member variable in the struct being assigned to:
* Passing and Returning Structs
  + Example struct was passed in, another local copy scoped just to the function was created, the local copy was modified and then returned, and lastly the value was read out.
  + Even though this works, remember that structs can be large - that is, they take up more memory than simple variables. Passing around large arrays will be much slower than passing a pointer to a struct into a function.

2.2 Advanced Compilation

* -Wall - By adding the "-Wall" switch, which turns on all warning notifications from gcc
* Clang - It indicates with a carat ^ where the particular issue occurs
* -c99 - In gcc, we can use command switches to specify which version of the C language we want to use.
* Makefiles - As compilation methods and command lines grow more complex, we eventually discover the need to build scripts that handle lengthy compilation jobs for us. To make things easier, a program called "make" reads "makefiles" that are essentially extremely configurable compilation scripts.
  + These makefiles can do things like compile programs together, deal with dependencies in compilation order, clean up compilation helper and temp files, compile in different configurations depending on command line switch, and much, much more.

2.2 Program Outlining in Program 2

* PseudoCoding Section

2.4 lseek() Dangers

* lseek() is a funny function. All can seem correct: your code will compile, and the program seems to execute.
* Make sure you're checking the man pages for the right include files

2.4 Manipulating Directories

* Creating a directory is very simple. We use the mkdir() system call and include the set of permissions we'd like the new directory to have: “”“int result = mkdir("newdir", 0755);”””
* If mkdir() returns 0, then the directory creation was successful.

To search through directories and files, we'll need several functions:

* opendir() - Opens up a directory so that it's contents can be analyzed
* readdir() - Return basic information on the next entry in the directory
* closedir() - Close down a directory that was previously opened
* stat() - Get detailed information about a file or directory