How to use make utility to build C projects?`

When we build projects in C/C++, we have dependencies among files. For example, there might be a file a.c that calls a function from b.c. So we must compile b.c before a.c.

There might be many dependencies in a project and manually following these dependencies and compiling files one by one becomes difficult. So, we are going to see how make utility can help us in making this simple.

First we need to create 4 files in which 2 of them are .c files and 1 header ( .h ) file and 1 make ( .mk ) file.

Let us name the files as client.c and server.c and server.h and makefile.mk

Makefile is a set of commands (similar to terminal commands) with variable names and targets to create object file and to remove them. In a single make file we can create multiple targets to compile and to remove object, binary files. You can compile your project (program) any number of times by using Makefile. The main idea of this file is to specify the dependencies.

make utility : This is a command line utility and used to process the instructions written in the Makefile.

Let us take a simple example.

The client.c file contains the main function, the server.c file contains the user defined function, 

The third file which is server.h header file which calls the user defined functions in the server.c file and the fourth file which is makefile.mk which contains set of all the commands with their variable names.

Here we write the client.c file

// This is client.c file

#include "stdio.h"

// This is header file that we have created

// in the beginning.

#include "server.h"

int main()

{

printf("hey there, welcome to ");

greetings();

return 0;

}

This is the client.c file which contains two header files one is #include”stdio.h” and the other is #include”server.h” file and remember this is the same file that we have created in the beginning and this contains main function which contains the printf statement which prints as “hey there, welcome to” (without quotes) and the main functions also calls the other user defined function which is greetings() .

Now we write the server.c file

// This is server.c file

#include "server.h"

#include "stdio.h"

void greetings()

{

printf("geeksforgeeks !");

}

In this server.c which includes the two header files one is #include”stdio.h” and the other is #include”server.h” file and remember this is the same file that we have created in the beginning and this contains user defined greetings function which contains the printf statement which prints as “geeksforgeeks !” (without quotes).

Now we write the server.h file

/ This is server.h file

void greetings();

This server.h file is very simple this calls the functions which are written in that file and when we include this header file to the other c programs then we can use the functions defined in this header file. Here this server.h file includes the functions wherever it is included.

Now we write the makefile.mk file

/// This is makefile.c file

a : client.o server.o gcc

client.o server.o client.o : client.o server.h gcc

- c client.c server.o : server.o server.h gcc

- c server.c

Now read this carefully here i will show you how to write makefile,

This is in windows so the object file is “a” if you are using the Linux, you replace “a” by “a.out”(without quotes)

**a : client.o server.o gcc**

See the first line of the code in that “a” represents the object file which contains all the code that we wrote till now, after “a” there are two more object file they are client.o and server.o these are the object files which are required to make the object file “a”,

client.o server.o client.o : client.o server.h gcc

In the next line there is gcc command remember this there should be 1 Tab space before writing the gcc command( if you forgot to put the tab this program will not run), the gcc command compiles the files given to it and stores in the name of its object files.

This is easily understood like here,  
target target\_name : prerequisites   
command with tab space

- c client.c server.o : server.o server.h gcc

Now lets move to the third line here the client.o is needed(because this is used in first line of code)so the pre-requisits for that file are client.o and server.h file the gcc command will compile the client.c to get the client.o file.

The last thing we need is server.o file to get that file we need server.o object file and the server.h header file

c server.c

the gcc compiler will compile the server.c file to get the server.o file   
now all things we need is ready, the makefile code is complete now.

Now will see how to run the make file.

*// This is used to run the makefile   
make -f makefile.mk*

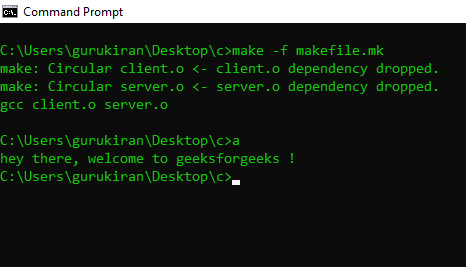
This is the syntax to run the makefile after typing this press enter the code will compile and that will create an executable file named “a” (in windows), “a.out”(in linux).

Now to execute the file it’s here

*// Remember this should be done only after the makefile command is executed   
// This in windows   
a   
//this in linux   
./a.out*

Examples:

Output : hey there, welcome to geeksforgeeks



How does make utility work internally? It creates a dependency graph of tasks and uses [Topological Sorting](https://www.geeksforgeeks.org/topological-sorting/) algorithm to find a valid sequence that follows all of the given dependencies.