Welcome back to the second lecture of E. C. E. two six four in the fall semester of 2020.

This lecture talks about make file.

Earlier, we talk about the three formats of C programs: the text format, the object format, and the executable format.

The text format is readable by humans; the object and the executable formats are readable by machines but not humans.

A complex C program may be created by using many text files. By using G. C. C. minus C, each dot C. file produces a corresponding dot O. . object file.

The process from a dot C file to a dot O file is called compilation.

The object files are then put together using the linking process to create an executable file.

To put this in another way, an executable file may be created by linking several object files. Each object file is created by compiling a text file.

What are the advantages of this two-stage process?

First, separating human readable files from machine readable files allow the same programs written by humans to run on different types of machines.

Second, if only one of the text file is modified, only the corresponding object file needs to be recreated. Since the other text files are unchanged. It is not necessary to recreate the object files.

The newly created object file can be linked with the unchanged object files to create the executable file.

Doing this can save time spent recreating the object files unnecessarily. A complex program may have hundreds of text files and hundreds of corresponding object files. If only one change is needed, this two-stage process save precious time, from several seconds to many minutes.

Let’s review what we have discussed so far about the two-stage process to convert the text files to an executable file.

First, we have to ensure that all the relevant G. C. C. warnings are turned on. There warnings have been shown to be very effective detecting mistakes.

We need to keep track which dot C. files have been changed since their corresponding objects files have been created.

If a dot C file has been changed, the corresponding object file needs to be generated. If a dot C file has not been changed recently, its corresponding object file does not need to be regenerated.

All the object files need to linked together to create the executable.

This is a lot of work if you have to do everything by hand.

Fortunately, a lot of work can be automated by using make file.

Make file, by convention, uses upper case M. and lower cases for the other letters.

Make file is a text file that tells Linux how to create an executable file from one or several text files.

Make file has different syntax from C programs. Comments start with the #, not slash slash.

The next three lines define symbols warning, error, and coverage. By convention, symbols use upper case letters. These symbols will be used later in the same make file.

The next line defines a new symbol for G C C and it uses the previously defined symbols. In make file, using a previously defined symbol, we need to use the dollar sign and parentheses, as shown here.

The next line defines the symbols for conditional compilation.

The bottom of this slide lists the source files. By convention, this uses the symbol S R C S.

.

These are the text files as parts of the C program. Please notice that these files’ names are separated by space, not commas.

Let’s scroll down.

Next, the symbol O B J S. lists the object files.

This line says “take the list of the files and replace the extension of dot C by the extension of dot O.”

.

In other words, O B J S means the list of A D D dot O, D I V dot O, M U L dot O, M A I N dot O, S O L U T I O N dot O, and S U B dot O.

.

The next line has a different structure. Instead of using an equal sign, this line uses colon. The left side of the colon is the target. The right side of the colon is the dependence. This line says the target is called M A I N and it depends on O B J S. .

What is O B J S?

.

It is define right above as the list of object files.

This line says that if the list of object files exist, then take the next line of action.

The line of action must start with a TAB key. Please remember that it is the TAB key, not space.

What do these two lines mean? They mean the following:

The files in the O B J S list must become available because they are on the right side of colon.

After these files become available, take the action in the second line. This line takes G C C, together with the test flags defined earlier and the list of object files, to create an executable file with the name of M A I N.

.

How do we know that this will create an executable file? Because the G C C command has no minus C.

.

Without minus C, G C C creates an executable file.

How do we know the executable is called M A I N?

.

Because minus O. is followed by the name of the executable.

.

The M A I N executable needs the object files. How are the object files created? These two lines compile C files to create object files.

These two lines say the following: If an object file is needed, find the corresponding C file and compile it.

How do we know the C file will be compiled? Because of the minus C option.

It is understandable that these symbols and syntax may look somewhat strange to you when you see them the first time.

Do not worry. You will soon get used to them.

Most people do not write Make file from scratch. Instead, most people take existing Make files and modify them.

You do not have to memorize all the details. You need to understand Make files so that you know what changes are needed.