Makefile - Recompilation

The **make** program is an intelligent utility and works based on the changes you do in your source files. If you have four files main.cpp, hello.cpp, factorial.cpp and functions.h, then all the remaining files are dependent on functions.h, and main.cpp is dependent on both hello.cpp and factorial.cpp. Hence if you make any changes in functions.h, then the **make** recompiles all the source files to generate new object files. However if you make any change in main.cpp, as this is not dependent of any other file, then only main.cpp file is recompiled, and help.cpp and factorial.cpp are not.

While compiling a file, the **make** checks its object file and compares the time stamps. If source file has a newer time stamp than the object file, then it generates new object file assuming that the source file has been changed.

Avoiding Recompilation

There may be a project consisting of thousands of files. Sometimes you may have changed a source file but you may not want to recompile all the files that depend on it. For example, suppose you add a macro or a declaration to a header file, on which the other files depend. Being conservative, **make** assumes that any change in the header file requires recompilation of all dependent files, but you know that they do not need recompilation and you would rather not waste your time waiting for them to compile.

If you anticipate the problem before changing the header file, you can use the `-t' flag. This flag tells **make** not to run the commands in the rules, but rather to mark the target up to date by changing its last-modification date. You need to follow this procedure −

- Use the command `make' to recompile the source files that really need recompilation.
- Make the changes in the header files.
- Use the command `make -t' to mark all the object files as up to date. The next time you run make, the changes in the header files do not cause any recompilation.

If you have already changed the header file at a time when some files do need recompilation, it is too late to do this. Instead, you can use the `-o file' flag, which marks a specified file as "old". This means, the file itself will not be remade, and nothing else will be remade on its account. you need to follow this procedure −

- Recompile the source files that need compilation for reasons independent of the particular header file, with `make -o header file'. If several header files are involved, use a separate `-o' option for each header file.
- Update all the object files with `make -t'.