Building makefiles for use with g++

Makefiles

The unix system has what is called a "make" utility. This allows the building of a configuration file to assist in compilation. Once the correct commands are placed in the makefile, the program can be rebuilt at any time with the command: "make".

There are many flags and parameters that can be used in makefiles, but here we will just present the basic format of the makefile.

First, the filename itself should be either "makefile" or "Makefile".

The makefile consists of sections, each of which specifies how to make a specific target. This target file is usually the name of an executable program or an object code file. The basic format of a section is:

```
<target_name>:  <dependency list>
        <commands>
```

The target name is a label that specifies what target this section is to make.

The dependency list is used to help the make utility decide whether or not the target needs rebuilding. Specifically, the entire first line of a section is saying "If anything in the dependency list has CHANGED since the last build, rebuild this target".

The <commands> part refers to the actual unix commands that are to be executed if this section needs rebuilding. This can consist of multiple commands (one per line), and each command must be preceded by a single "tab" character.

Any line that starts with a # character is a comment.

Sample makefile:

```
# This is a comment line
# Sample makefile for fraction class

frac: main.o frac.o
	g++ -o frac main.o frac.o

main.o: main.cpp frac.h
	g++ -c main.cpp

frac.o: frac.cpp frac.h
	g++ -c frac.cpp
```

```
clean:
	rm *.o frac
```

The first section specifies "frac" as the target, and it depends on main.o and frac.o. If either of these files changed since the last build, then "frac" must be rebuilt. The command is the linking command for linking these two object code files together into a target executable called "frac":

```
	g++ -o frac main.o frac.o
```

The next section specifies how to built the target "main.o". This depends on main.cpp and frac.h (if either file changes, main.o must be rebuilt). Similarly, the next section does the same for the target "frac.o". The commands for these sections are the normal g++ commands for the compile stage (i.e. compile source code into object code):

```
	g++ -c main.cpp
	g++ -c frac.cpp
```

Any section can be invoked specifically with the command:

```
	make <target_name>
```

For instance, to build only the "frac.o" target, use:

```
	make frac.o
```

When the make command is used by itself, the utility attempts to make the FIRST target listed in the makefile -- this is why we list the executable program, "frac", first. So, using this makefile, the following two commands are equivalent:

```
	make frac
	make
```

The last section is a special line, being used to clean up files in the directory, rather than build anything. The target name is "clean", and the command is the remove command ("rm"). This target will remove the object code files and the executable from the current directory. This part of the makefile is invoked with the command:

```
	make clean
```