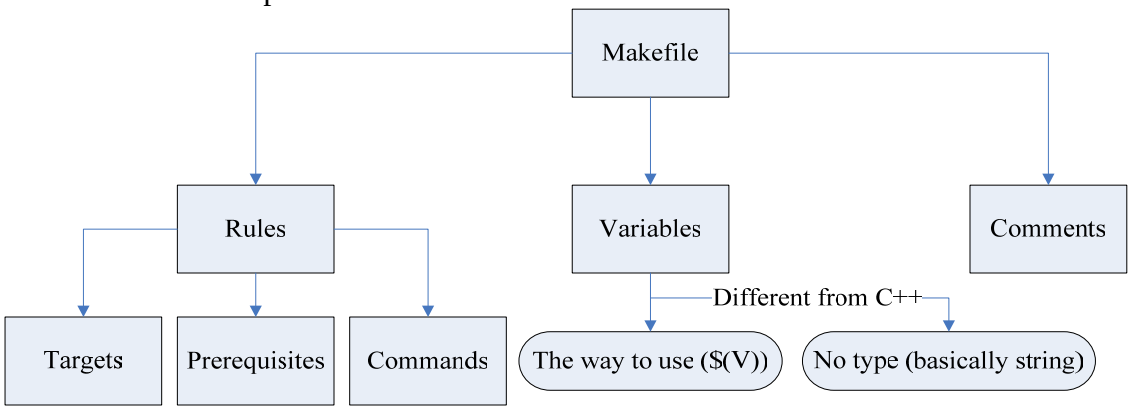
An Example of MakeFile

```
# comment on here
# To generate the executable, type “make all” or “make” at the command prompt.
# To remove the executable and any files, type “make clean” at the command prompt.
# MakeFile = comments+variables+rules

VARIABLE = some variables here
# The compiler and compiler options
CXX=g++
CXXFLAGS=-g -W -Wall -Werror -pedantic -ansi -I /usr/csshare/include
# libraries to use
LIBS = -lX11 –lccc
# where to look for libraries
LIBDIRS=-L/usr/X11R6/lib -L/usr/local/lib -L /usr/csshare/lib

# write some rules here (Remember the “tab” after the prerequisite before the command)
# "all" means we want everything including the executable and back up files after the
compilation
# "clean" will delete all the unneeded files like the executable etc. you may use this target
before turn in your project.
# rules = target + prerequisites + commands
# Prerequisites are a space separated list of things that must exist or be done before before
# any of the commands in the rule are executed. The items in the list of prerequisites can
# be file names, or the targets of other rules.

# compile the program
all: ccc_win_main.cpp
        $(CXX) $(CXXFLAGS) $(LIBDIRS) -o outputFileName ccc_win_main.cpp
$(LIBS)
# remove unnecessary files
clean:
        rm -rf *~ outputFileName
```



Structure of Makefile

A list of tags that is commonly used for g++

g++

This is the name of the compiler program.

-g

Leave debugging information in the executeable. This allows a programmer to use a debugging tool to step through the program code one line at a time and watch how variables change. This quarter we will learn how to use a debugging tool.

-Wall

The -Wall option tells the compiler, “If I do anything in this program that is really odd, warn me.” C++ has a long long list of things that could possibly indicate a programming mistake, but by default many of them are not checked during compilation. Adding -Wall ensures that many of the more complex things it can warn you about will be reported. (It is like having a very picky set of eyes checking over your program for you.)

-W

Not all warnings are turned on by the -Wall option. Yeah, it seems kinda redundant, but that's the way it is. The -W option tells the compiler, “I'd like you to warn me about all of the simple mistakes that people might make. ” This lets g++ do some of the checking for you that would otherwise cause you lots of time and trouble. (Remember, warnings that are caught are often logic errors that you don't have to fight! This is a ***very*** good thing.)

-Werror

The -Werror option is related to the other options in that they all deal with warnings. Unlike the others, it doesn't enable additional warning messages. Instead, -Werror tells the compiler, “If you ever feel like warning me about anything, consider that an error and do not compile any further.” This way even if you wanted to ignore the warnings, you would have to find a way to convince the compiler that you really know what you are doing.

-pedantic

The -pedantic option tells the compiler to use only standard features of the language and not to allow any non-standard C++ features that the compiler may allow.

-ansi

The -ansi option tells the compiler to only allow ANSI complient C++.

-o Writer

Write the output the the file “Writer”. If you do not use the -o option, the output will be written to a file named “a.out”.