HowTo Write Generic Makefiles

C++ (http://www .puxan.com/web/tag/cplusplus/) Open source (http://www .puxan.com/web/tag/open-source/)

Pattern rule to build objects

The first thing we need to write a generic Makefile is a rule to build object files without having to specify their name. That is to say, we need a pattern rule.

```
%.o: %.cpp
```

Now that we have a pattern rule, we will need to write the actions to be taken, and so we will need the file names. As we don't know them, because the rule is the same for all the **cpp** files, we are going to use **automatic variables**. This feature let us access to the current target name and dependencies.

There are many **automatic variables** but we will only need these two:

$@ Contains the target file name.
$< Contains the first dependency file name.

So a generic rule to compile source files could be as follows:

```
# Generic rule
%.o: %.cpp
        $(CC) -c $(CC_FLAGS) $< -o $@
```

Main target

Now that we know how to build the objects, we need to build the executable. To link executable we need all the object files. So we have to write a rule where the prerequisites are all the object files. To do that we could think of using something like ***.o**, but remember that object files don't yet exist. So we have to use source code names to obtain the object names. If we had all the sources names in **$SOURCES** we could do as follows:

```
OBJECTS = $(SOURCES:.cpp=.o)
```

To initialize **$SOURCES** do not use ***.cpp**, because **make** won't do what you expect. **Make** would compile all the sources, but they would all be compiled even if they haven't been modified. Therefore you would be loosing the power of this tool. To avoid this problem you can initialize **$SOURCES** like this:

```
SOURCES = $(wildcard *.cpp)
```

Generic example

Using the things previously explained we would be able to write a generic make. But notice that we haven't used header files in the dependencies part. Therefore if we modify a **header** file, make may not recompile all the code that depends on this header. So, after modifying a header file, we should type **make clean** before executing **make**.

```
# Declaration of variables
CC = g++
CC_FLAGS = -w

# File names
EXEC = run
SOURCES = $(wildcard *.cpp)
OBJECTS = $(SOURCES:.cpp=.o)

# Main target
$(EXEC): $(OBJECTS)
        $(CC) $(OBJECTS) -o $(EXEC)

# To obtain object files
%.o: %.cpp
        $(CC) -c $(CC_FLAGS) $< -o $@

# To remove generated files
clean:
        rm -f $(EXEC) $(OBJECTS)
```

See also

HowTo Write Simple Makefiles (/web/howto-write-simple-makefiles/)

References

GNU Make user (http://www.gnu.org/software/make/manual/make.html)