Defining Custom Suffix Rules in Makefile

Make can automatically create a.o file, using cc -c on the corresponding .c file. These rules are built-in the **make**, and you can take this advantage to shorten your Makefile. If you indicate just the .h files in the dependency line of the Makefile on which the current target is dependent on, **make** will know that the corresponding .cfile is already required. You do not have to include the command for the compiler.

This reduces the Makefile further, as shown below −

```
OBJECTS = main.o hello.o factorial.o
hello: $(OBJECTS)
   cc $(OBJECTS) -o hello
hellp.o: functions.h

main.o: functions.h
factorial.o: functions.h
```

Make uses a special target, named *.SUFFIXES* to allow you to define your own suffixes. For example, refer the dependency line given below −

```
.SUFFIXES: .foo .bar
```

It informs **make** that you will be using these special suffixes to make your own rules.

Similar to how **make** already knows how to make a *.o* file from a *.c* file, you can define rules in the following manner −

```
.foo.bar:
   tr '[A-Z][a-z]' '[N-Z][A-M][n-z][a-m]' < $< > $@
.c.o:
   $(CC) $(CFLAGS) -c $<
```

The first rule allows you to create a *.bar* file from a *.foo* file. It basically scrambles the file. The second rule is the default rule used by **make** to create a *.o* file from a *.c* file.