CSI 402 – Systems Programming
Another Example for Make

Handout 1.4

Note: This handout shows a `makefile` for the program shown in Handout 1.2.

File: `makefile`

```
#The following rule tells make about possible suffixes
#(extensions) of file names.

.SUFFIXES: .c .o

#The following definition of CC ensures that
#gcc will be used to compile the C source files.

CC = gcc

#The following definition of CFLAGS ensures that
#the debugger can be used with the executable file (sample)
#created by running make.

CFLAGS = -g

#The following rule tells make how a ".o" file should
#be created from the corresponding ".c" file.
#Note that the "-c" option must be used here since we are
#compiling source files separately. (Note that the line
#following the ".c.o:" line begins with the "tab" character.)

.c.o:
        $(CC) $(CFLAGS) -c $<

#Dependency rule for the default target and how the
#default target is to be created. (Note that the line
#following the dependency rule begins with the "tab"
#character.)

prog:   main.o  funct.o
        gcc  main.o  funct.o  -o  prog

#Dependency rules for other targets. (We don't need to
#specify how these targets are created since we have already
#given a general rule for creating a ".o" file from the
#corresponding ".c" file.)

main.o:   constants.h  struct_def.h  globals.h  prototypes.h
funct.o:  constants.h  struct_def.h  externs.h  prototypes.h

#Target for removing unnecessary files.
clean:
        rm -f *.o core
```