Makefile Explanation

A "Makefile" is a text file created in the project directory. By convention an upper case "M" is used. The general idea of a Makefile is to compile all the .c source code files to object files and then link these together to create the binary executable.

The Makefile for building the GTK4 application called "demo" is shown below.

```
CC ?= gcc
PKGCONFIG = $(shell which pkg-config)
CFLAGS = $(shell $(PKGCONFIG) --cflags gtk4)
LIBS = $(shell $(PKGCONFIG) --libs gtk4)
SRC = main.c
OBJS = $(SRC:.c=.o)
all: demo
%.o: %.c
	$(CC) -c -o $(@F) $(CFLAGS) $<
demo: $(OBJS)
	$(CC) -o $(@F) $(OBJS) $(LIBS)
clean:
	rm -f $(OBJS)
	rm -f demo
install:
	echo "Installing is not supported"
run:
	./demo
```

A bunch of variables called *CC, PKGCONFIG, CFLAGS, LIBS, SRC* and *OBJS* are created and can be accessed using the $ symbol. The *CC* variable is used to define the C compiler which in this case is gcc. The *PKGCONFIG* variable uses the shell "*which pkg-config*" to determine where the "*pkg-config*" utility is located on the system. The "pkg-config" utility is a helper tool used when compiling applications and tells the compiler where to find the GTK4 header and library files. *PKGCONFIG* is used to load up the *CFLAGS* and *LIBS* variables.

The SRC variable is used to store the source files to be compiled which in this case is a single file called "main.c". In other applications there will be more than one C file and the added files are separated with a space.

The OBJS objects variable specifies all the object files required to construct the program and is a list of all object file names. The OBJS objects variable is assigned ${SRC:.c=.o} which means take the variable value ${SRC}, which is usually a string composed of the source files separated by spaces and, for each file replace the suffix .c with .o (representing object files).

A Makefile contains targets and rules to build an application. The CC, *CFLAGS* and *LIBS* variables are used to compile a target called "demo" which is the program name. In order to write a target the name of the the target is written, then the prerequisites and then, on the next line, the rule indented using the "tab" key. It is important to use the "tab" key spacing.

The % symbol is used as a wild card substitution and allows pattern rules to be written using automatic variables which are variables for which the value is generated by Make, rather than having to be set explicitly. The pattern rule below uses the automatic variables @F and $<.

```
%.o: %.c
	$(CC) -c -o $(@F) $(CFLAGS) $<
```

This pattern rule can be interpreted as follows: to build <filename>.o, which depends on <filename.c>, use the command *$(CC) -c -o $(@F) $(CFLAGS) $<* Here the *$(@F)* and *$<* automatic variables evaluate to the target of the rule and the first prerequisite of the rule respectively. See the external links below for a more detailed explanation of pattern rules.

A target called “clean” has been added to the Makefile which will delete the executable and any object files so that a fresh build can be performed using the source code files which remain untouched. Running “make clean” at the command line removes the object file and the executable.

A target called “run” has also been added to the Makefile. The demo application can be run from the terminal using “make run”.

External Links

An Introduction to Makefiles
https://www.gnu.org/software/make/manual/html_node/Introduction.html

GNU Make automatic variables
https://www.gnu.org/software/make/manual/html_node/Automatic-Variables.html

Features of GNU make
https://web.mit.edu/gnu/doc/html/make_12.html