Makefile Tutorial for C Projects

What is a Makefile?

A **Makefile** is a special file used to control the build process of a C project. It defines how to compile and link the program using the `make` utility. It helps:

- Automate compilation
- Avoid redundant compilation
- Manage dependencies
- Handle multiple targets

Basic Makefile Structure

```
target: dependencies
\tcommand
```

- `target`: output file name - `dependencies`: source/header files needed - `command`: shell command (MUST be indented by a TAB)

Example 1: Single File Program

File: hello.c

```
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Makefile

```
hello: hello.c
\tgcc hello.c -o hello
```

Usage:

```
make         # builds hello
./hello      # runs the program
```

Example 2: Multi-File Program

Files: main.c, maths.c, maths.h

main.c

```
#include <stdio.h>
#include "maths.h"

int main() {
    int sum = add(5, 7);
    printf("Sum: %d\n", sum);
    return 0;
}
```

maths.c

```
int add(int a, int b) {
    return a + b;
}
```

maths.h

```
#ifndef MATHS_H
#define MATHS_H

int add(int a, int b);

#endif
```

Makefile

```
CC = gcc
CFLAGS = -Wall -g
OBJ = main.o maths.o
TARGET = main

$(TARGET): $(OBJ)
\t$(CC) $(CFLAGS) -o $(TARGET) $(OBJ)

main.o: main.c maths.h
maths.o: maths.c maths.h

clean:
\trm -f *.o $(TARGET)
```

Example 3: Pattern Rules

```
CC = gcc
CFLAGS = -Wall -g
TARGET = main
SRCS = main.c maths.c
OBJS = $(SRCS:.c=.o)

$(TARGET): $(OBJS)
\t$(CC) $(CFLAGS) -o $@ $^

%.o: %.c
\t$(CC) $(CFLAGS) -c $<

clean:
\trm -f $(TARGET) $(OBJS)
```

Example 4: Project Directory

Structure:

```
project/
          src/
                main.c
                maths.c
          include/
                maths.h
          Makefile
```

Makefile:

```
CC = gcc
CFLAGS = -Iinclude -Wall -g
SRCS = src/main.c src/maths.c
OBJS = $(SRCS:.c=.o)
TARGET = bin/main

$(TARGET): $(OBJS)
\t$(CC) $(CFLAGS) -o $@ $^

src/%.o: src/%.c
\t$(CC) $(CFLAGS) -c $< -o $@

clean:
\trm -f src/*.o $(TARGET)
```

Best Practices

- Use variables for compiler and flags
- Separate source, header, and object files
- Always include a clean target
- Use pattern rules for scalability

Advanced Tips

Phony Targets

```
.PHONY: clean all run
```

Automatic Dependency Generation

```
-include $(SRCS:.c=.d)
```

Command to Generate .d files:

```
gcc -MMD -c file.c
```

Build Commands

```
make          # Build target
make clean    # Remove binaries and object files
make          # Rebuild everything
```