GCC and CMake Tutorial with Static Libraries

1 Project Structure

We will use the following files:

```
project/
           main.c
           calculations.c
           calculations.h
```

2 Source Code

2.1 Header (calculations.h)

```c
#ifndef CALCULATIONS_H
#define CALCULATIONS_H

double add(double a, double b);
double multiply(double a, double b);
double power(double base, double exp);

#endif
```

2.2 Implementation (calculations.c)

```c
#include <math.h>
#include "calculations.h"

double add(double a, double b) {
    return a + b;
}

double multiply(double a, double b) {
    return a * b;
}

double power(double base, double exp) {
    return pow(base, exp);  // requires -lm
}
```

2.3 Main Program (main.c)

```c
#include <stdio.h>
#include "calculations.h"

// Example: macro can be set with gcc -DSCALE=10
#ifndef SCALE
#define SCALE 1.0
#endif

int main() {
    double x = 2.0, y = 3.0;

    printf("Add: %f\n", add(x, y) * SCALE);
    printf("Multiply: %f\n", multiply(x, y) * SCALE);
    printf("Power: %f\n", power(x, y) * SCALE);

    return 0;
}
```

3 Using GCC Directly

3.1 Compile and Link in One Step

```
gcc -DSCALE=10 main.c calculations.c -o main -lm
./main
```

3.2 Separate Compilation

```
# Compile object files
gcc -c calculations.c -o calculations.o
gcc -c -DSCALE=5 main.c -o main.o

# Link together with math library
gcc main.o calculations.o -o main -lm
./main
```

3.3 Create a Static Library (.a)

```
# Compile object file
gcc -c calculations.c -o calculations.o

# Create static library
ar rcs libcalculations.a calculations.o

# Link main with the static library
gcc -c -DSCALE=2 main.c -o main.o
gcc main.o -L. -lcalculations -o main -lm

./main
```

4 Using CMake

4.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 3.10)
project(GccTutorial C)

# Create the static library
add_library(calculations STATIC calculations.c)

# Build the executable
add_executable(main main.c)

# Link math library and calculations
target_link_libraries(main calculations m)

# Pass SCALE as a definition
target_compile_definitions(main PRIVATE SCALE=10)
```

4.2 Build with CMake

```
mkdir build && cd build
cmake ..
cmake --build .
./main
```

4.3 Override SCALE at Configure Time

```
cmake -DSCALE=20 ..
cmake --build .
./main
```

5 Conclusion

This tutorial demonstrated:

- Using `gcc` with defines and the math library.
- Creating object files, executables, and static libraries (`.a`).
- Using CMake to manage the same workflow in a scalable way.