Tut

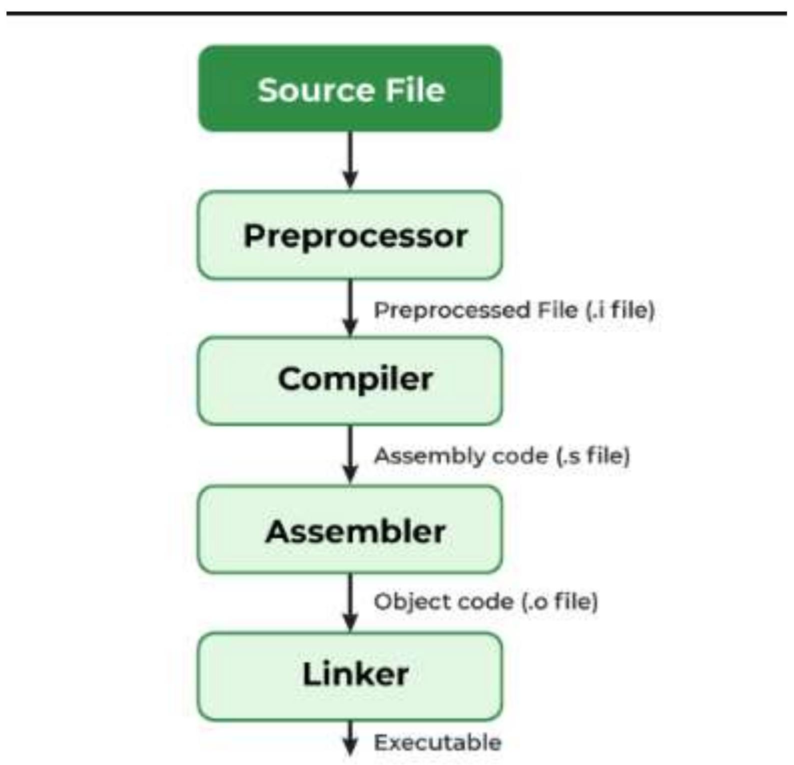
Date - 23-10-24

TA - Harshvardhan

What is Multi-File Programming?

- In C programming, multi-file programming refers to organizing your code across multiple files, which can help improve the modularity, readability, and maintainability of a project.
- Instead of writing all the code in one large file, you split it into separate files with each file responsible for a specific part of the program (e.g., function definitions, data structures, or global variables).
- The general practice is to separate your code into **header files (.h)** and **source files (.c)**.

Compilation Process (Revisit)



Link - https://www.geeksforgeeks.org/compiling-a-c-program-behind-the-scenes/

Header File (math_ops.h)

here

i) #ifndef MATH_OPS_H

ii) #define MATH_OPS_H

iii) int add(int a, int b);

iv) int subtract(int a, int b);

v) #endif

Why the Macros ?

The #ifndef, #define, and #endif directives are include guards to prevent multiple inclusions of the same header file

Function Declaration (math_ops.c)

```
// math_ops.c
vi) #include "math_ops.h"

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

Viii) int subtract(int a, int b) {
          return a - b;
        }
```

Main Function (main.c)

```
ix)     #include <stdio.h> //Include  -- stdio.o
x)      #include "math_ops.h“ //
xi)     #include "math_ops.h“ (again)

xi)   int main() {
xii)  int result1 = add(10, 5);
xiii) int result2 = subtract(10, 5);

xiv) printf("Addition: %d\n", result1);
xv)  printf("Subtraction: %d\n", result2);

xvi) return 0;
      }
```

Now how to compile ?

```
gcc main.c math_ops.c -o program
```

Breakup of what is happening ?

```
(gcc -c math_ops.c
gcc -c main.c
gcc -o program main.o math_ops.o)
```

Advantages

- **Modularity**: Each module (or functionality) of your program is isolated into its own file, making it easier to maintain and develop.
- **Reusability**: You can reuse the same code across different projects by simply including the relevant header and source files.
- **Teamwork**: Different team members can work on different parts of the project without interfering with each other.
- **Faster Compilation**: When making changes, only the modified source files need to be recompiled, not the entire program.

Make

```
CC = gcc
# Compiler flags
CFLAGS = -Wall -Wextra -std=c99
# Target executable name
TARGET = program
# Source files
SRCS = main.c math_ops.c
# Object files (derived from source files)
OBJS = $(SRCS:.c=.o)
# Default target to build the executable
$(TARGET): $(OBJS)
$(CC) $(CFLAGS) -o $(TARGET) $(OBJS)
# Rule to compile .c files into .o files
%.o: %.c
$(CC) $(CFLAGS) -c $< -o $@
# Clean up object files and the executable
.PHONY: clean
clean:
rm -f $(OBJS) $(TARGET)
```

File I/O

Code Demo