# Preprocessors in C

•••

Derrick

# Header File

- Extension : .h
- Function declarations and macro definitions to be shared between source files.
- Including a header file is equal to copying the content of header file.
- A simple practice in C is that we keep all constants, macros, system wide global variables, and function prototypes in the header file.

# Preprocessors

The C Preprocessor is not a part of the compiler, but is a separate step in the compilation process. In simpler terms, a C Preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation.

All preprocessor commands begin with a hash symbol (#).

# Directives (Macros)

- #define : substitutes a preprocessor macro
- #include : inserts a particular header from another file
- #undef : undefines a preprocessor macro
- #ifdef : returns true if this macro is defined
- #ifndef : returns true if this macro is not defined
- #if : tests if a compile time condition is true
- #else
- #elif
- #endif
- #error : prints error message on stderr
- #pragma : Issues special commands to the compiler, using a standardized method.

# Predefined Macros

- __DATE__ : the current date as a character literal in “MMM DD YYYY” format
- __TIME__ : the current time as a character literal in “HH:MM:SS” format
- __FILE__ : this contains the current filename as a string literal
- __LINE__ : this contains the current line number as a decimal constant
- __STDC__ : defined as 1 when the compiler compiles with the ANSI standard.

# Command line commands (Preprocessing)

```
gcc -E -C -P -D{{arg}} main.c
```

```
-E : preprocess

-C : include comments

-P : pretty (get rid of #-line)

-D : pass arg as #define arg
```

# Compile / Link

*main.c*     *multiply.c*     *multiply.h*

To compile,

```
gcc -c main.c
gcc -c multiply.c
```

To simply link object files,

```
gcc main.o multiply.o -o outputFileName
```

To link and compile,

```
gcc main.c multiply.c -o outputFileName
```