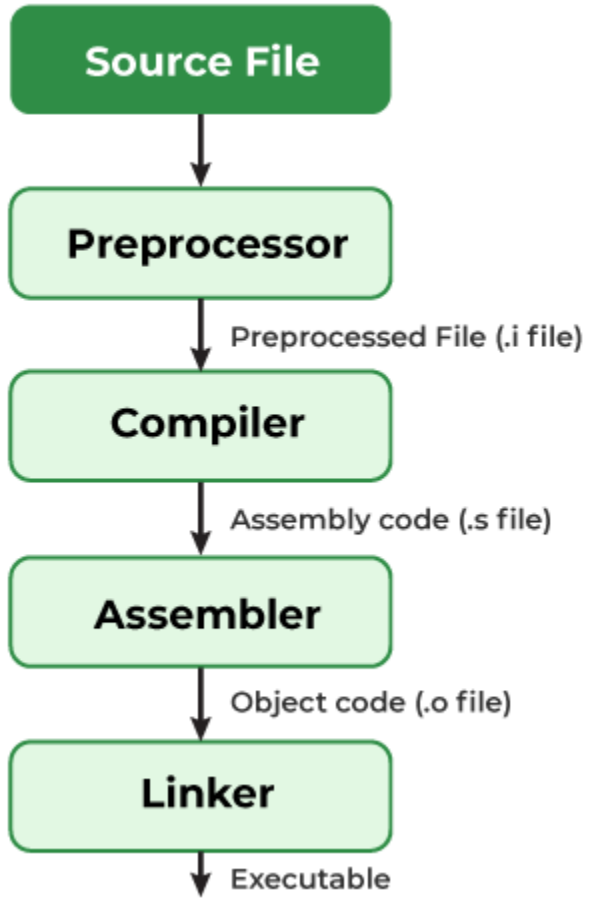
# Preprocessor


Jinwoo Choi

# Preprocessor(전처리기)

**Preprocessors** are programs that **process the source code before the actual compilation begins**. It is not the part of compilation, but a separate process that allows programmers to modify the code before compilation. It is the first process that the C source code goes through while being converted to executable file.



# Preprocessor(전처리기)

| Operator | Description |
|---|---|
| #define | Used to define a macro |
| #undef | Used to undefined a macro |
| #include | Used to include a file in the source code program |
| #ifdef | Used to include a section of code if a certain macro is defined by #define |
| #ifndef | Used to include a section of code if a certain macro is not defined by #define |
| #if | Check for the specified condition |
| #else | Alternate code that executes when #if fails |
| #elif | Combines else and if for another condition check |
| #endif | Used to mark the end of #if, #ifdef, and #ifndef |

# #define

```c
#include <stdio.h>

#define MAXLOOP 3

#define MUL(a, b) (a * b)

int main(void)
{
    int x = 9;
    int y = 11;

    printf("%d\n", MUL(x, y));

    for(int i = 0; i < MAXLOOP; i++)
    {

        printf("loop: %d\n", i);
    }

    return 0;
}
```

```
jinwoo@DESKTOP-UEN32NR:~$ ./a.out
99
loop: 0
loop: 1
loop: 2
```

to see the result of preprocessing(gcc –E)

```
jinwoo@DESKTOP-UEN32NR:~$ gcc -E test.c
```

```c
int main()
{
    int x = 9;
    int y = 11;

    printf("%d₩n", (x * y));

    for(int i = 0; i < 3; i++)
    {

        printf("loop: %d₩n", i);
    }

    return 0;
}
```

# #undef

```c
#include <stdio.h>

#define MUL(a, b) (a * b)

int main(void)
{
    int x = 9;
    int y = 11;

#undef MUL
    printf("%d\n", MUL(x, y));

    return 0;
}
```

```
jinwoo@DESKTOP-UEN32NR:~$ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -Wextra -Werror test.c
test.c: In function ‘main’:
test.c:11:20: error: implicit declaration of function ‘MUL’ [-Wimplicit-function-declaration]
   11 |     printf("%d\n", MUL(x, y));
      |                     ^~~
```

# #ifdef/else/endif

gcc -D*MACRO*=*VALUE source.c*

ex) gcc –DVERSION=3 –DRELEASE test.c

```c
#include <stdio.h>

int main(void)
{
#ifdef DEBUG
    int x = 9;
    int y = 11;
#else
    int x = 6;
    int y = 3;
#endif
    printf("%d\n", x * y);

    return 0;
}
```

```
jinwoo@DESKTOP-UEN32NR:~$ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -Wextra -Werror test.c
jinwoo@DESKTOP-UEN32NR:~$ ./a.out
18
jinwoo@DESKTOP-UEN32NR:~$ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -Wextra -Werror test.c -DDEBUG
jinwoo@DESKTOP-UEN32NR:~$ ./a.out
99
```

# #if/elif/else/endif

gcc -D*MACRO*=*VALUE source.c*

ex) gcc –DVERSION=3 –DRELEASE test.c

```
jinwoo@DESKTOP-UEN32NR:~$ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -Wextra -Werror test.c
jinwoo@DESKTOP-UEN32NR:~$ ./a.out
99
jinwoo@DESKTOP-UEN32NR:~$ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -Wextra -Werror test.c -DMODE=2
jinwoo@DESKTOP-UEN32NR:~$ ./a.out
18
jinwoo@DESKTOP-UEN32NR:~$ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -Wextra -Werror test.c -DMODE=999
jinwoo@DESKTOP-UEN32NR:~$ ./a.out
408
```

```c
#include <stdio.h>

#ifndef MODE
#define MODE 1
#endif

int main(void)
{

#if MODE == 1

    int x = 9;
    int y = 11;

#elif MODE == 2

    int x = 6;
    int y = 3;

#else

    int x = 12;
    int y = 34;

#endif

    printf("%d\n", x * y);

    return 0;
}
```

# LAB – Preprocessor

- Create a file named 'Preprocessor_YourName.c'.
- Your program should
  - Print a basic startup message(e.g., "program started")
  - Use #ifdef to conditionally display extra debug messages(try other things too)
  - Always print a final message(e.g., "program ended")
- Compile the program with and without the macro
- **gcc –DMACRO Preprocessor_YourName.c**
- **gcc  Preprocessor_YourName.c**