

Topic 5: Preprocessor

Preprocessor

- Ask the compiler to perform some works in advance
- Common preprocessor
 - #define
 - #include
 - #ifdef #ifndef #if #else #elif #endif
 - The usage

```
#ifdef DEBUG_INFO
printf("debug: %d \n", x);
#endif
```

Preprocessor

- `#ifdef (DEBUG_INFO)` equals to `#if defined (DEBUG_INFO)`
 - The usage of `#if defined` is more complicated
 - Example
 - `#if !defined DEBUG_INFO`
(If not defining `DEBUG_INFO`, the process goes)
 - `#if defined DEBUG_INFO || !defined VERBOSE_INFO`
(Both two condition should satisfy)
 - `#if 0` (mark a whole segment)
 - `#if VERBOSE_INFO > 2`
 - `#if chip == INTEL`
- Define the compile flag during compiler time → `gcc -DDEBUG_INFO`

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

```
int main(void)
```

```
{
```

```
#ifdef Linux
```

```
    printf("LINUX\n");
```

```
#elif defined (Windows)
```

```
    printf("Microsoft Windows\n");
```

```
#elif defined(OS)
```

```
    printf("OS=%s", OS);
```

```
#else
```

```
    printf("Unknown\n");
```

```
#endif
```

```
}
```

```
> gcc define.c
```

```
> ./a.out
```

```
Unknown
```

```
> gcc -DWindows define.c
```

```
> ./a.out
```

```
Microsoft Windows
```

```
> gcc -DOS=\"Sun\" define.c
```

```
> ./a.out
```

```
OS=Sun
```

Preprocessor

- In a large-scale C project, the header file .h contains

```
#ifndef _GLOBE_H
```

```
#define _GLOBE_H
```

```
typedef ...
```

```
... ..
```

```
... ..
```

```
#endif /*GLOBE_H*/
```

Assume that both a1.c and a2.c include this header file. When a1.c was compiled earlier, this preprocessor can avoid duplicate declaration.

Preprocessor

- If you find **multiple definition of xxxxx** when building your project
 - Check if you add the `#ifndef`
 - Check the compilation unit
 - Two object file `.o` include common objects
 - → solution: (1) Use the **extern** keyword in the header `.h` file
(2) Declare the main body in a C file

<https://www.unix.com/programming/219335-c-program-multiple-definition-error-during-linking-time.html>

The new GCC (at least mine) seems to handle this kind of simple error directly

#pragma

- The keywords for compiler → configure the compiler for cross platform

- Example

#pragma asm: The following parts are assembly language (Microsoft C)

#pragma small: Small memory mode (Microsoft C)

#pragma code: Put read only data in ROM to save RAM (Keil C)

- Can be use as message

```
#ifdef _X86
```

```
#pragma message("_X86 macro activated!")
```

```
#endif
```

Reference: <http://topalan.pixnet.net/blog/post/22334686>

inline (can taken as macro)

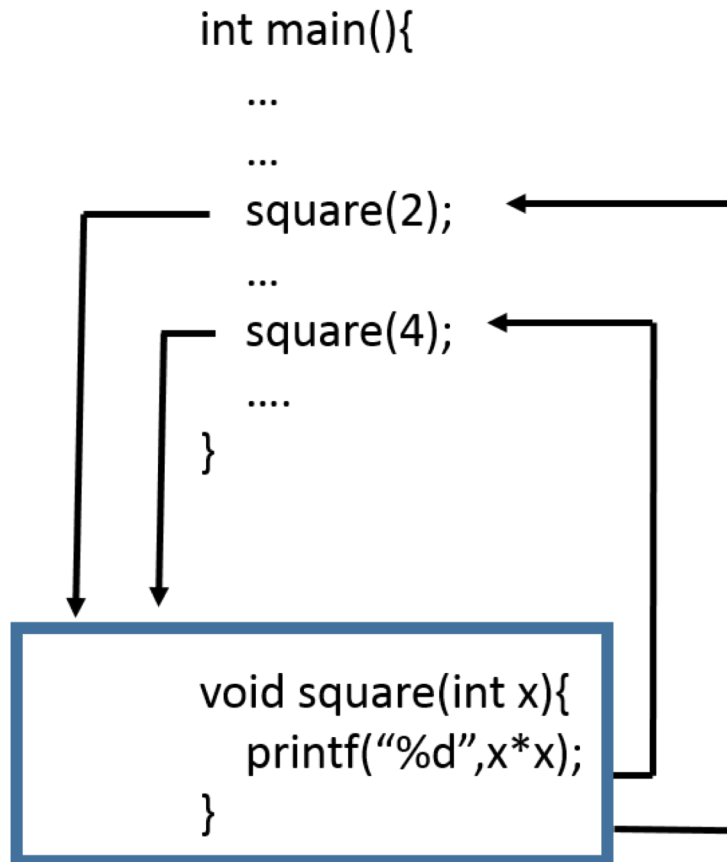
- Ask the compiler to set a function as an “inline function”
- Ex:

```
inline void set_bit (Int8 *target, Int8 bit)
{ .....

}
```

If the compiler accepts your demand, the above function can be taken as

```
#define set_bit xxxx
```

General procedure

```
int main(){
    ...
    ...
    {
        x =2;
        printf("%d",2*2);
    }
    ...
    {
        x =4;
        printf("%d",4*4);
    }
    ....
}
```

Inline function

Special compiler keywords

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

```
#pragma message("Compiling") //not support in new gcc version
```

```
int main(void)
```

```
{  
    printf("Compiling %s, line: %d, on %s, at %s, STDC=%d \n",  
          __FILE__, __LINE__, __DATE__, __TIME__, __STDC__);  
    //__STDC__ : check if ANSI C  
    return 0;  
}
```

```
#if __STDC__  
extern int a1(int);  
#else  
extern int a1(void);  
#endif
```

Makefile

CC = gcc

OBJ = main.o change.o

EXE = run

all: \$(EXE)

.c.o: ; \$(CC) -c \$*.c

\$(EXE): \$(OBJ)

\$(CC) -o \$@ \$(OBJ)

clean:

rm -rf \$(EXE) *.o *.d core

build all .c files to be .o files

\$@ means include the parameter \$(EXE)

You should use “TAB” to indent

W10-on site assignment

- Divide your assignment last week to be four files main.c, ui.c, ui.h, list.c, list.h, **you should have your own makefile**
 - ui.c → include those functions about user interface
 - list.c → **includes list operations**



```
C list.c
C list.h
C main.c
M makefile
C ui.c
C ui.h
```

main.c

```
1  #include "ui.h"
2  #include "list.h"
3
4  int main (void)
5  {
6      tNumStorHead *list;
7
8      list = initial_list();
9      while (1)
10     {
11         ui_main_menu(list);
12     }
13 }
14
```

ui.h

```
1  #ifndef __UI_H__
2  #define __UI_H__
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  #include "list.h"
8
9  void ui_main_menu( .... );
10
11 void ui_add_menu( .... );
12 void ui_get_add_location( .... );
13
14 void ui_del_menu( .... );
15 int ui_get_del_location( .... );
16
17 #endif
```

list.h

```
1  #ifndef __LIST_H__
2  #define __LIST_H__
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  typedef struct num_storage
8  {
9      int number;
10     struct num_storage *next;
11     struct num_storage *prev;
12 } tNumStorage;
13
14 typedef struct num_stor_head
15 {
16     int counts;
17     struct num_storage *head;
18     struct num_storage *tail;
19 } tNumStorHead;
20
21 tNumStorHead* initial_list( .... );
22 void list_print( .... );
23 void list_add_node( .... );
24 void list_del_node( .... );
25
26 #endif
```

printf is not allowed in
initial_list()
list_add_node()
list_del_node()