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

```c
#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)

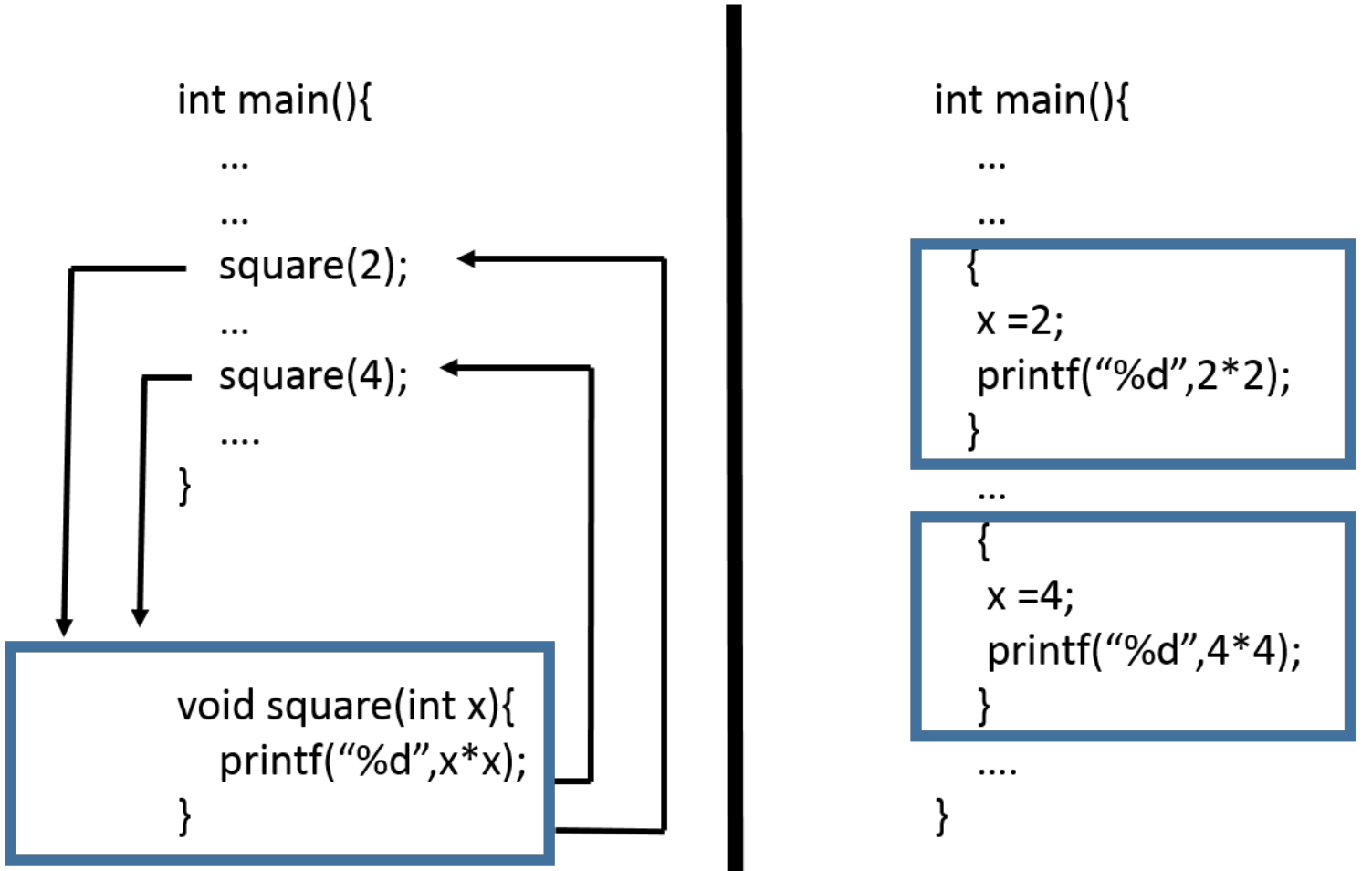
- Ask the compiler to set a function as an “inline function”
- Ex:

```
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

Inline function

Special compiler keywords

```
#include <stdio.h>

#pragma message(“Compiling”)  //not really when using gcc

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

W9-on site assignment

- Divide your assignment last week to be four files main.c, main.h, list.c, list.h, you should have your own makefile
  - main.c includes main, process_add, process_delete functions
  - main.h includes function prototypes in main.c
  - list.c includes the operations and corresponding functions about lists
  - list.h includes list head and list node declarations and function prototypes in list.c

- Requirement about includes
  - In main.h, you can include list.h
  - In main.c, you can only include main.h
  - In list.c, you can only include list.h

```c
#include "main.h"

int main (void)
{
    tNumStorHead *list;
    list = (tNumStorHead *) malloc (sizeof(tNumStorHead));
    initial_list(list);
    get_input(list);
}

int get_input(tNumStorHead *list)
{
    int choice;
    while (1)
    {
        printf ("1. Add a number or 2. Delete a number: ");
        scanf ("%d", &choice);
        if (choice == 1)
        {
            process_add(list);
        }
        else if (choice == 2)
        {
            process_delete(list);
        }
        else
        {
            printf ("  No such choice !\n");
            continue;
        }
        printf ("\n");
    }
}
```

```c
void process_add(tNumStorHead *list)
{
    int number, location, choice;

    process_add_get_input(list->counts, &number, &location, &choice);
    add_node_to_list(list, number, location, choice);
    print_list(list);
}
```

Separate selection by different functions