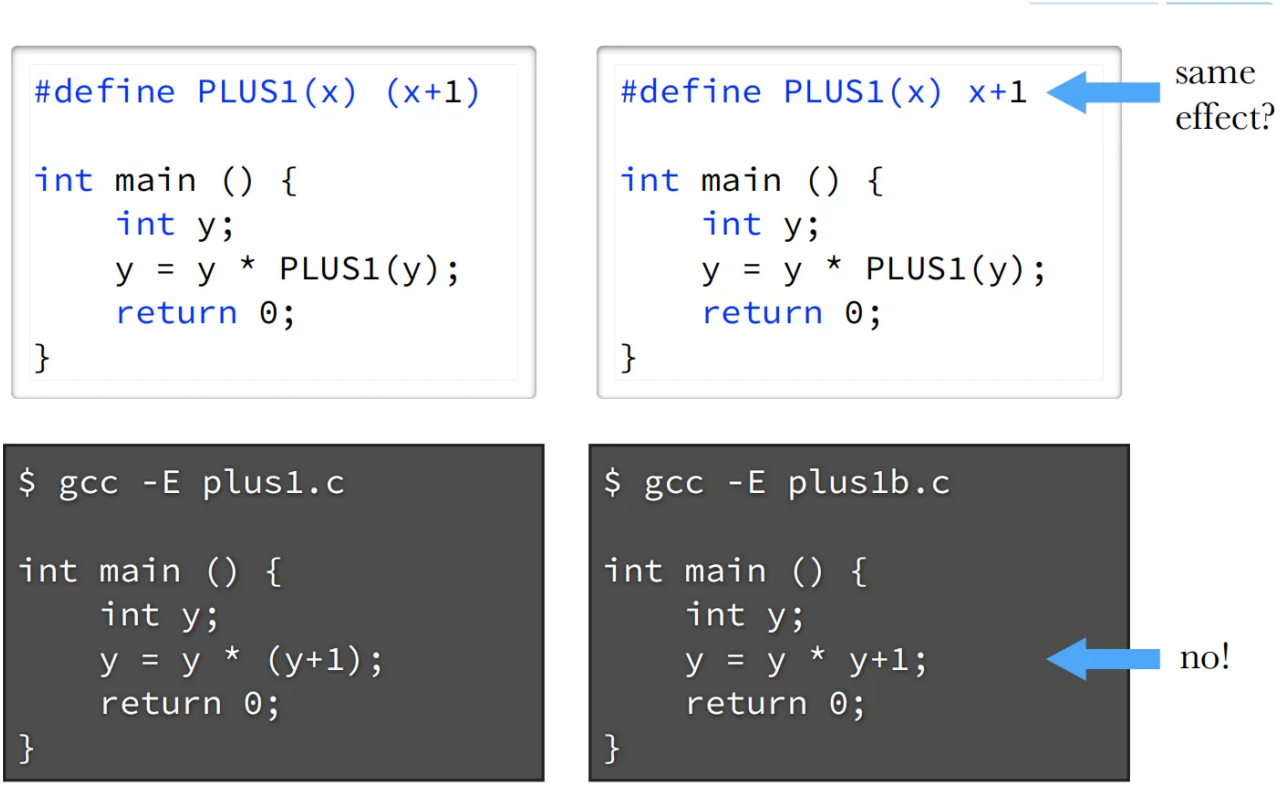
第三课

gcc 预处理（preprocessing）命令

1．-E：只进行预处理，输出预处理结果

预处理不会展示头文字

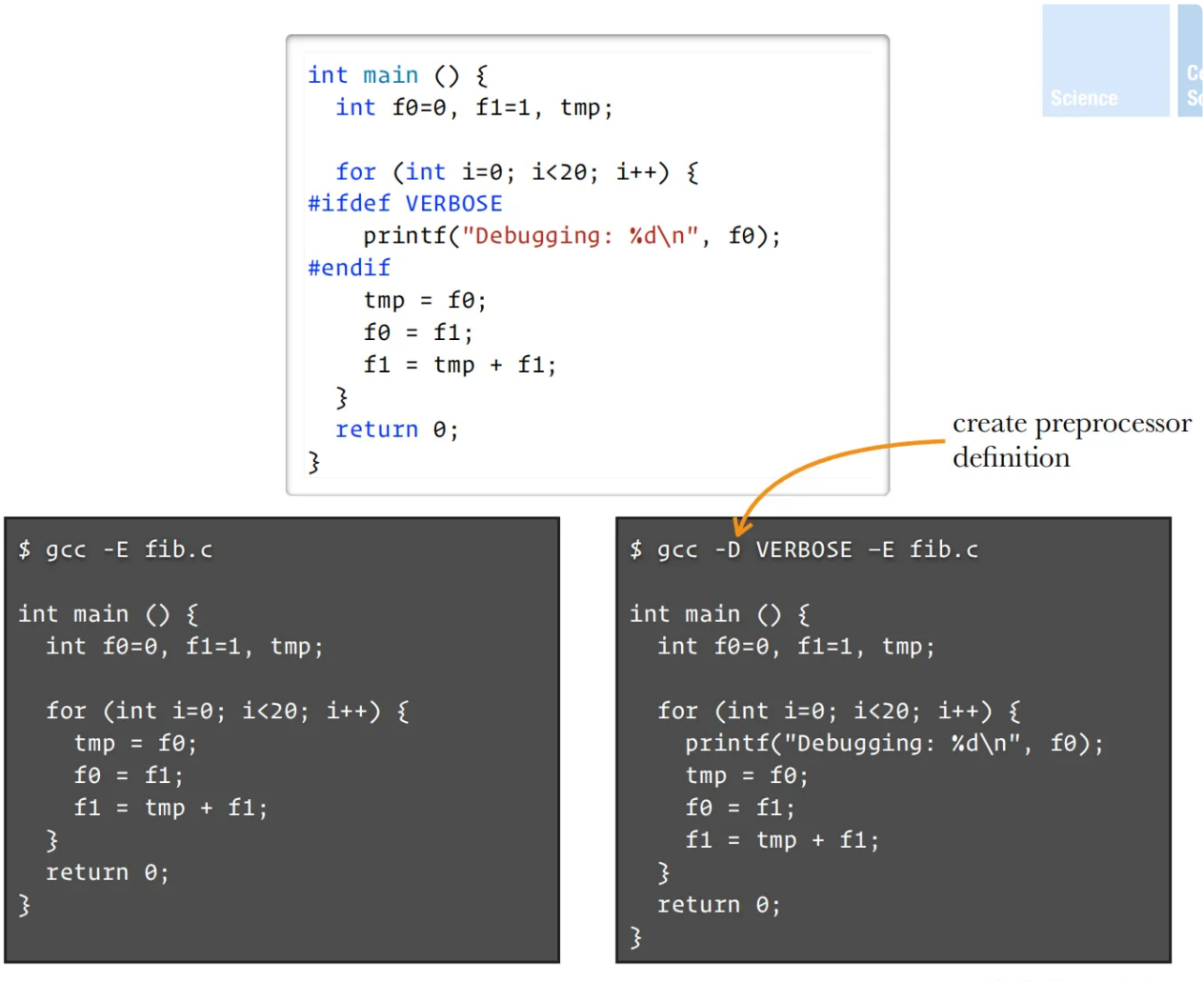
> 注意预处理时会发生宏定义替代
>
> 所有宏定义函数需要加括号（如下图）



2．`-D<macro>`：定义预处理宏

多用于 debugging，当我想 debugging 时，使用-D 添加预处理宏定义，开启 debugging。

比如：



3. 非链接编译

`-c`：只编译源文件，生成目标文件（`.o` 文件），不进行链接

这个和第二节课讲的-o 连接存在区别，即：不会链接头文件

4. -o 链接的一些问题

4.1. 变量识别问题

连接器无法识别被定义在两个不同文件中变量，需要使用 extern 声明变量（extern 表示如果你在函数内找不到变量，则往外层寻找）：

错误：

sum.c

```c
int sumWithI(int x, int y) {
    int I;
    return x + y + I;
}
```

main.c

```c
#include <stdio.h>

int sumWithI(int, int);

int I = 10;

int main() {
    printf("%d\n", sumWithI(1, 2));
    return 0;
}
```

```
$ gcc -Wall -o demo sum.c main.c
$ ./demo
-1073743741
```

正确：

sum.c

```c
int sumWithI(int x, int y) {
    extern int I;
    return x + y + I;
}
```

main.c

```c
#include <stdio.h>

int sumWithI(int, int);

int I = 10;

int main() {
    printf("%d\n", sumWithI(1, 2));
    return 0;
}
```

```
$ gcc -Wall -o demo sum.c main.c
$ ./demo
13
```

4.2. 变量寻找问题

被 static 定义的变量或函数，只能在单一文件中可用/可见，且只会被定义一次（初始化赋值操作也只会进行一次，哪怕函数被多次调用，函数中的 static 变量也不会被重新初始化赋值）

在 B 文件中使用 static 定义了一个变量，当与 A 头文件进行连接时，会找不到该变量（static 出现在 A 文件中也会出现同样的问题）。

sum.c

```c
int sumWithI(int x, int y) {
    extern int I;
    return x + y + I;
}
```

main.c

```c
#include <stdio.h>

int sumWithI(int, int);

static int I = 10;

int main() {
    printf("%d\n", sumWithI(1, 2));
    return 0;
}
```

```
$ gcc -Wall -o demo sum.c main.c
Undefined symbols:
  "_I", referenced from:
      _sumWithI in ccmvi0RF.o
ld: symbol(s) not found
collect2: ld returned 1 exit status
```

Linux 下 objdump 命令的使用

> 课上用的不多，简单看一下

objdump 工具是用来显示二进制文件的信息，就是以一种可阅读的格式让你更多地了解二进制文件可能带有的附加信息。该命令常用于 Linux下反汇编目标文件或者可执行文件。