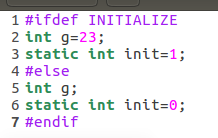
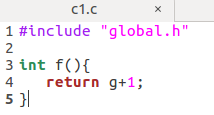
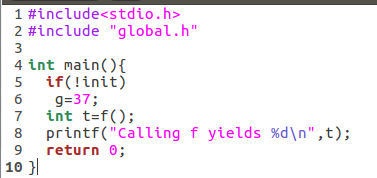
计算机组成作业

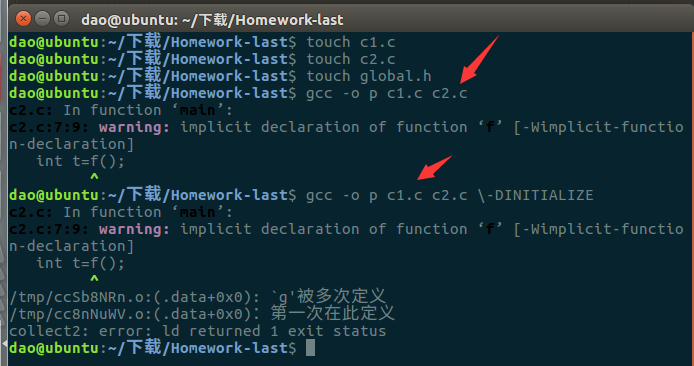
软件1401

宗玉芬-24

第七章：

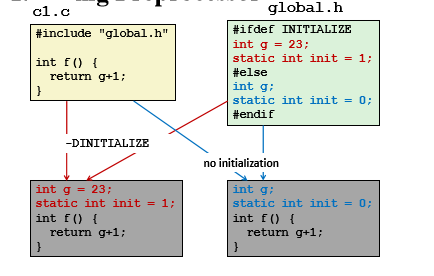


根据题意编译c1.c和c2.c后出现如下情况



下面进行分析

两种情况的过程如图所示不同的编译命令时得到不同的c1.c，从而c2.c也不同



1. **Gcc -o p c1.c c2.c**

报出警告，但结果和我们预期一样如下图为38。其中警告是因为使用了外部函数f()但是没有申明。第一种情况比较好理解

C1.c:

int g;

static int init = 0;

int f() {

return g + 1;

}

C2.c:  
#include <stdio.h>

int g;

static int init = 0;

int main() {

if (!init) {

g = 37;

}

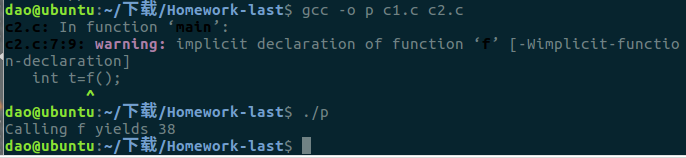
int t = f();

printf("calling f yields %d\n", t);

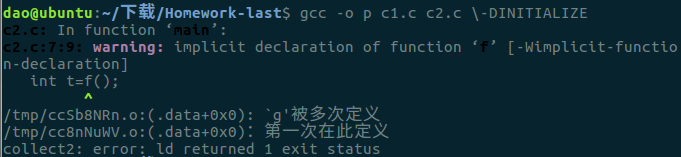
return 0;

}

从代码可知两个弱符号时从中任意选择一个，则不会报错



2. **Gcc -o p c1.c c2.c \-DINITIALIZE**



错误的原因是在DINITIALIZE时代码如下。

C1.c:

int g=23;

static int init = 1;

int f() {

return g + 1;

}

C2.c:  
#include <stdio.h>

int g=23;

static int init = 1;

int main() {

if (!init) {

g = 37;

}

int t = f();

printf("calling f yields %d\n", t);

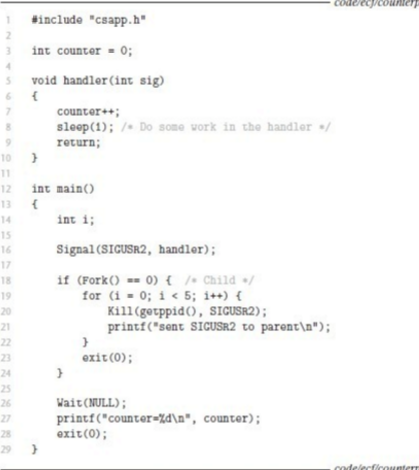
return 0;

}

强符号g被多次定义，违背了链接器处理多重定义符号的第一条规则：不允许有多个强符号

第八章：

8.23．



之所以counter的值总是2是因为在任何时刻，一种类型至多只会有一个待处理信号。当子进程发送第一个信号给父进程时，父进程进入handler将counter++并阻塞了SICUSR2,接着第二个信号发送则被挂起称为待处理信号，排队等待接收。往后的信号不会排队等待而是被简单的丢弃。所以counter的值总是2。