

操作系统原理实验报告

**实验名称:** 实验五 内核线程

**授课教师：** 张青

**学生姓名:**

**学生学号:** 2233

1. **实验要求**





1. **实验过程**

（1）Assignment 1

通过学习函数内部引用可用参数列表中的参数，定义一个具有可变参数的函数print\_any\_number\_of\_integers,用于输出若干个整数的函数。测试此函数是否可正常执行。

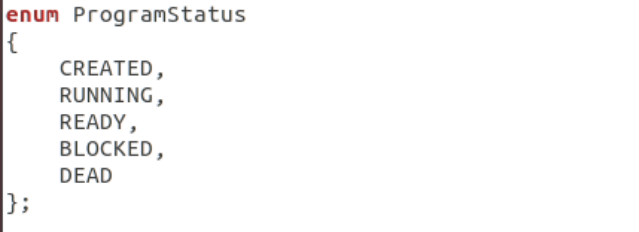
然后实现可变参数机制，此处我们使用实现的宏来引用可变参数，同样测试函数是否可正常执行。

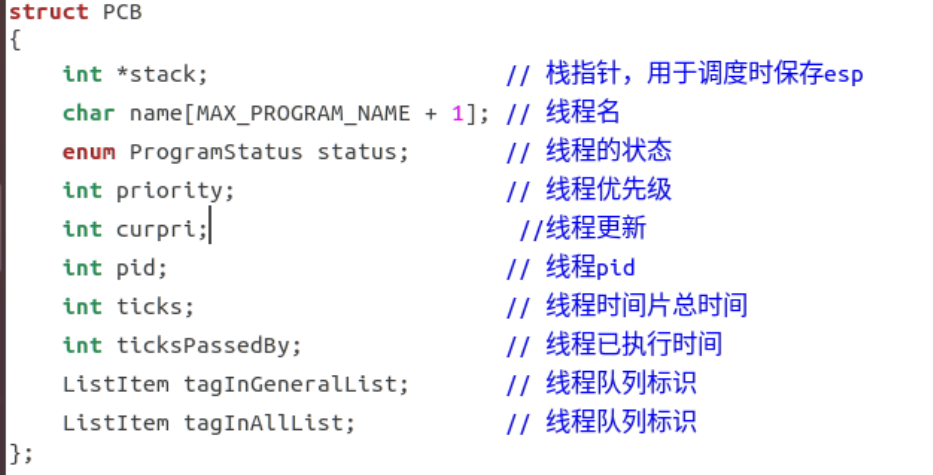
实现一个能够输出字符串的函数STDIO::print，接下来再编写printf函数，用于解析相应参数: 如果遇到普通字符，则直接输出； 如果遇到%，则看%的下一个字符是什么。

如果 % 的下一个字符是'\0'，则退出；如果是'c'，则以字符的格式返回一个可变参数列表中的参数，并将其加入缓冲区；如果是's'，则输出并清空缓冲区，并以 char\*的格式返回一个可变参数列表中的参数，并直接打印；如果是'd'，则以 int 的格式返回一个可变参数列表中的参数，转换为字符串后加入缓冲区；如果是'x'，则以 int 的格式返回一个可变参数列表中的参数，进行进制转换，然后转换为字符串，加入缓冲区，如果是‘f’，则以 double 的格式返回一个可变参数列表中的参数，转换为字符串后加入缓冲区，按照示例步骤一步步编写即可实现一个printf函数，测试函数得到相应实验结果。

（2）Assignment 2

设计PCB并实现线程，首先需要创建线程的五个状态并声明PCB的结构：



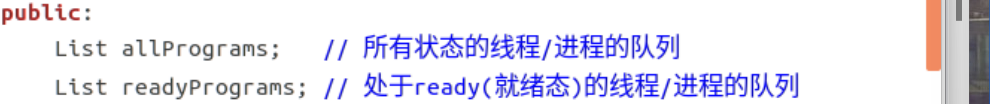


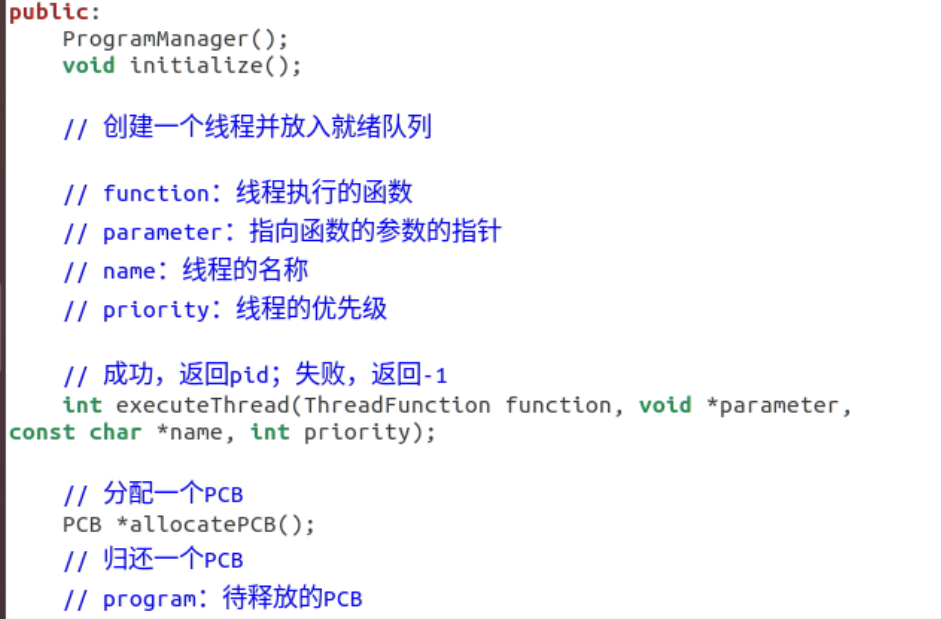
此处相较于示例代码添加了curpri表示cur\_priority,用于调整优先级，防止实现某些调度算法时的饿死现象。

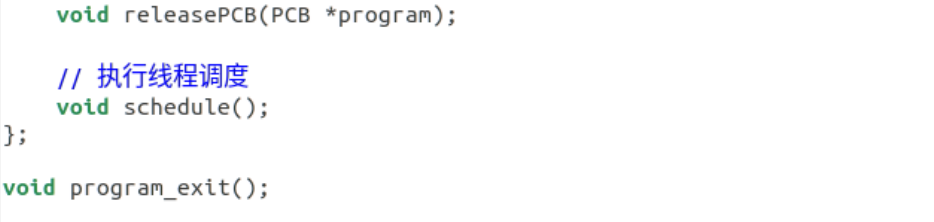
声明了list类为带头结点的双向列表。

实现一个程序管理类，用于进行线程和进程的创建和管理：

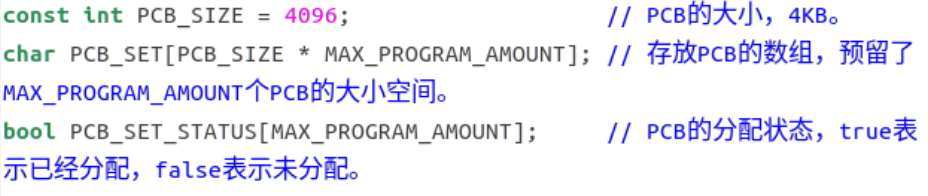








以此为目录依次实现相对应函数，注意在创建线程前我们需要先向内存申请一个PCB，此实验中我们的PCB大小是4096个字节：



在使用我们实现的程序管理类的成员函数之前，我们首先需要初始化类，编写函数为programManager::initialize。

而我们注意到PCB分配工作需要线程互斥，故我们此处采用简单的关中断和开中断来实现线程互斥：



依据以上逐一实现相应函数，完成线程创建，测试结果。

（3）Assignment 3

由于此前和后续未实际调用任务二添加的curpri，故此处删去不作考虑，编译并进入调试界面，

执行以下命令：

a. b c\_time\_interrupt\_handler

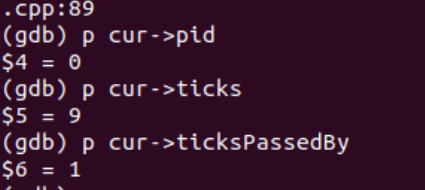
b. c

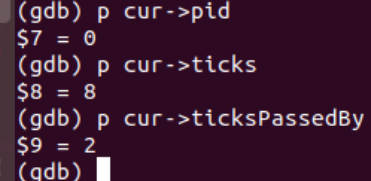
c. p cur ->pid

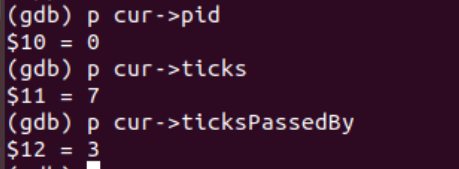
d. p cur ->ticks

e. p cur ->ticksPassedBy

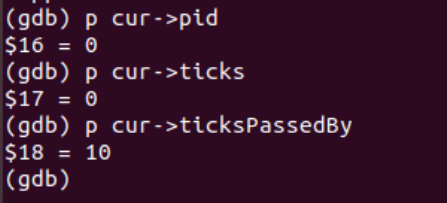
重复b-e的操作步骤，我们可以看到，ticks在递减，ticksPassedBy在递增，最后ticks递减到0：







…………



于是我们可以知道因为我们此时采用时间片轮转算法，每创建新进程都会被分配一个时间片，加入CPU运行队列。每时间中断一次，时间片递减一次，时间片记录增加一次，当分配时间被耗尽时会进行调度操作。

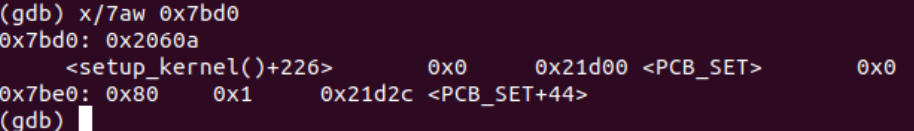
时间片等于0后，我们设置第二个断点asm\_switch\_thread,并循环进行以下操作，观察esp存储的内容：

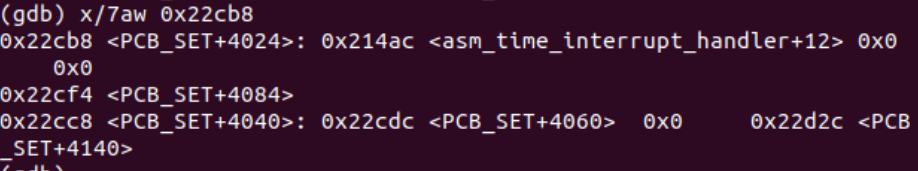
a.c

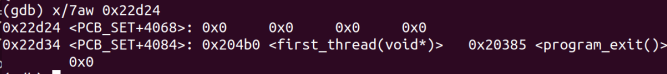
b. info registers得到esp地址

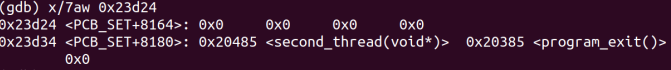
c. x/7aw esp 地址

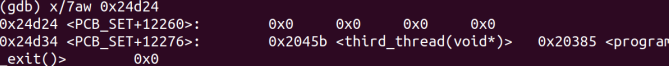
线程切换即将线程栈指针放到esp中的过程，于是我们可以通过观察esp中内容来理解线程调度过程：



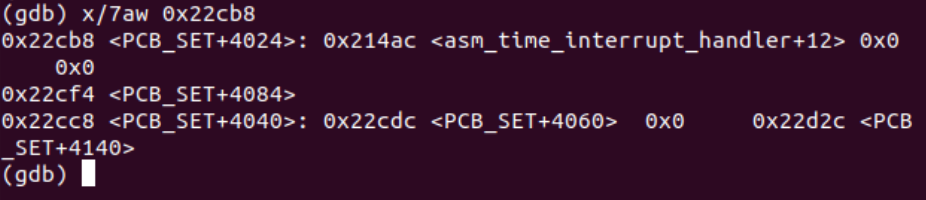








执行完最后一个线程后：



从上述过程可以较为直观的看出系统会保存准备被换下线程的上下文，包括栈、寄存器、 PC 等。 下次被换上后，会恢复该线程，并且 CPU 的 pc 跳转到原来线程 pc 所指的位置继续运行。

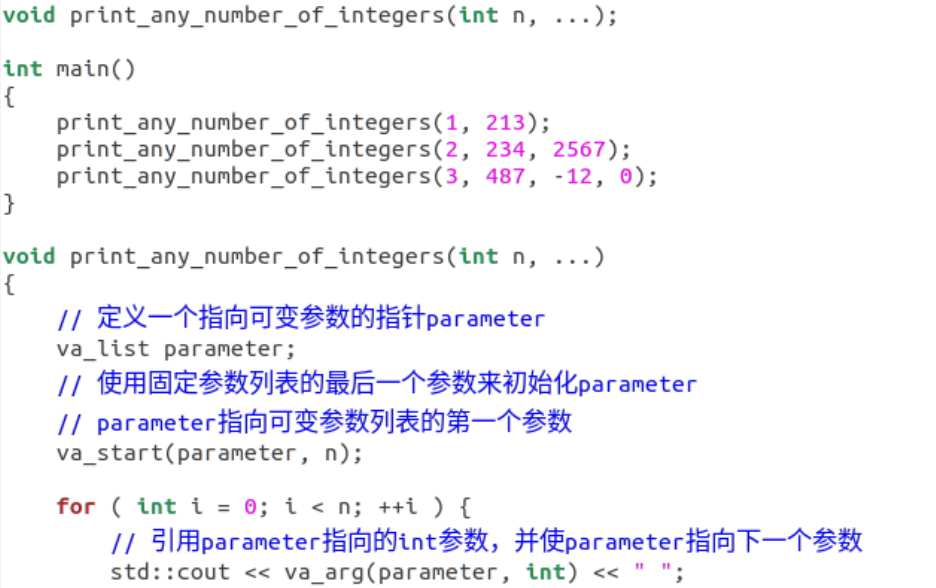
（4）Assignment 4

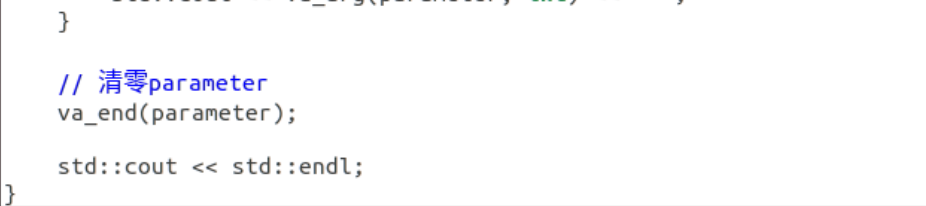
此处将任务二代码中的时间片轮转算法更改为先来先服务算法，相应改动的函数详见关键代码处，编译执行可得实验结果。

1. **关键代码**

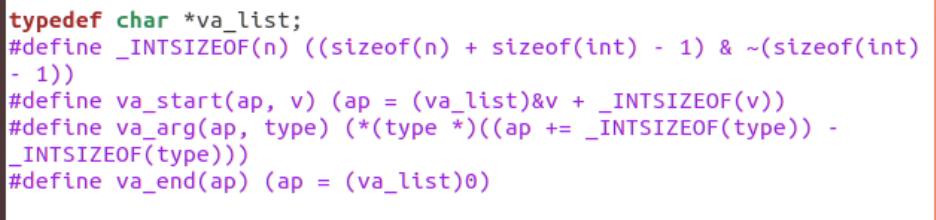
（1）Assignment 1

可变参数的函数：

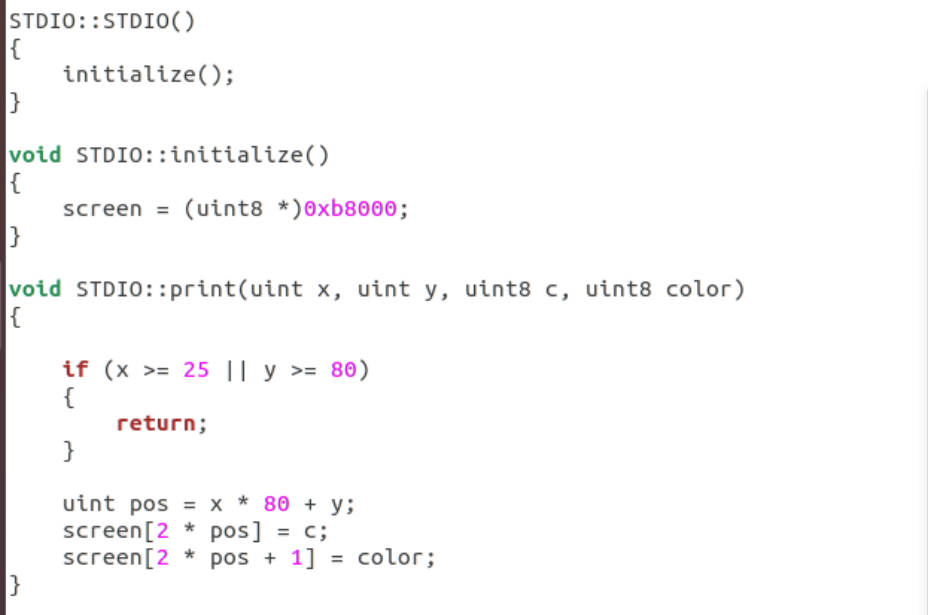


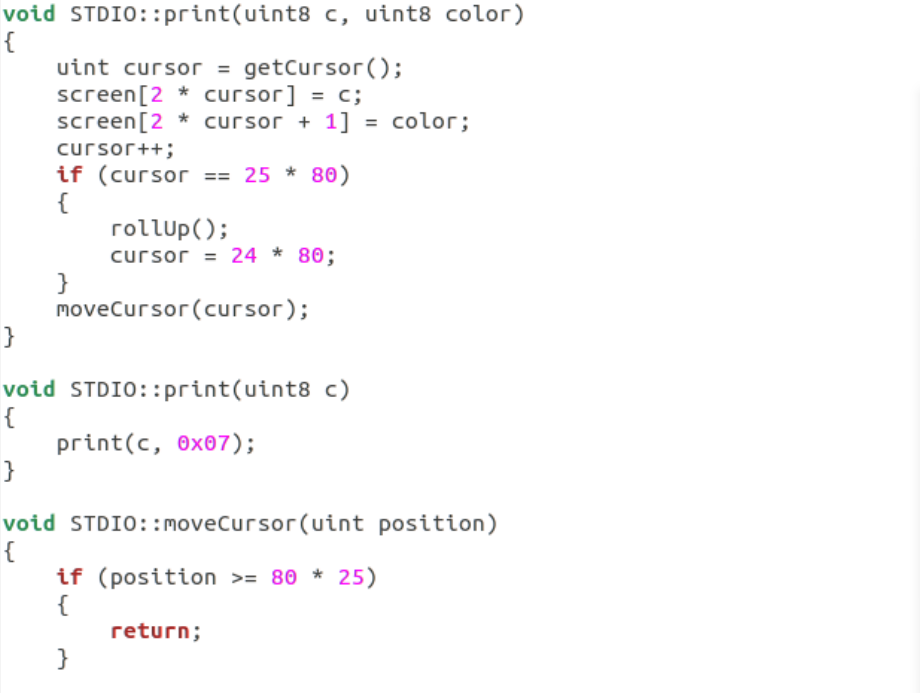


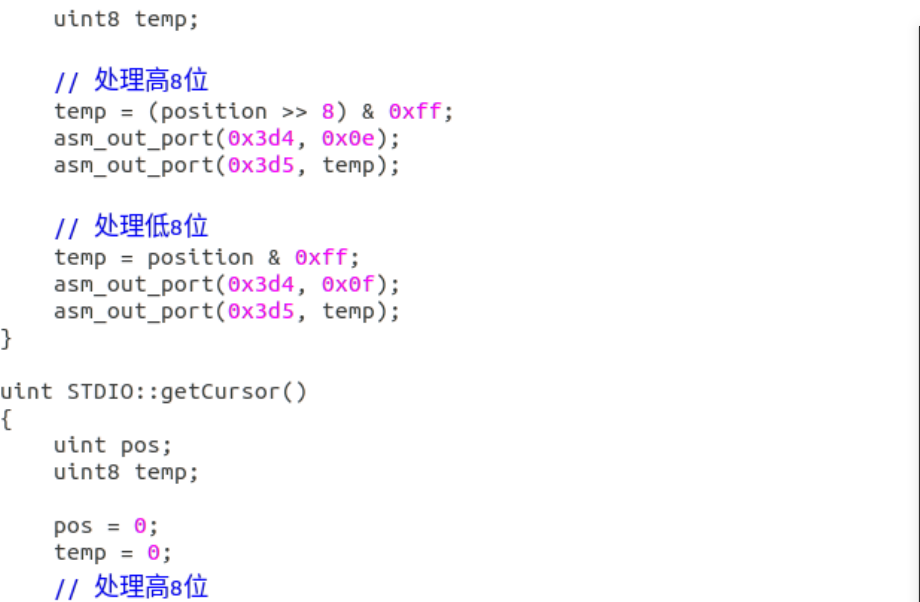
可变参数实现机制：

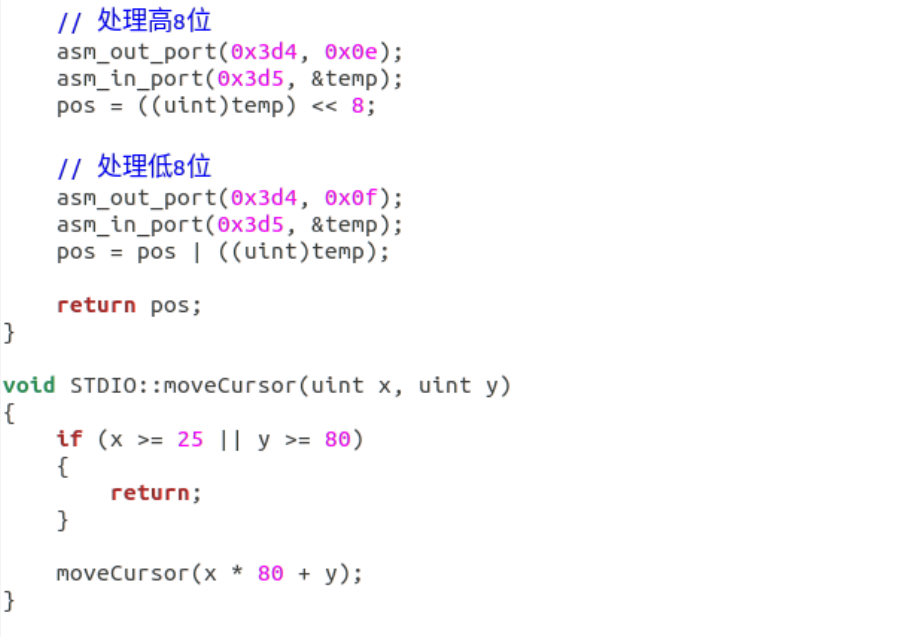


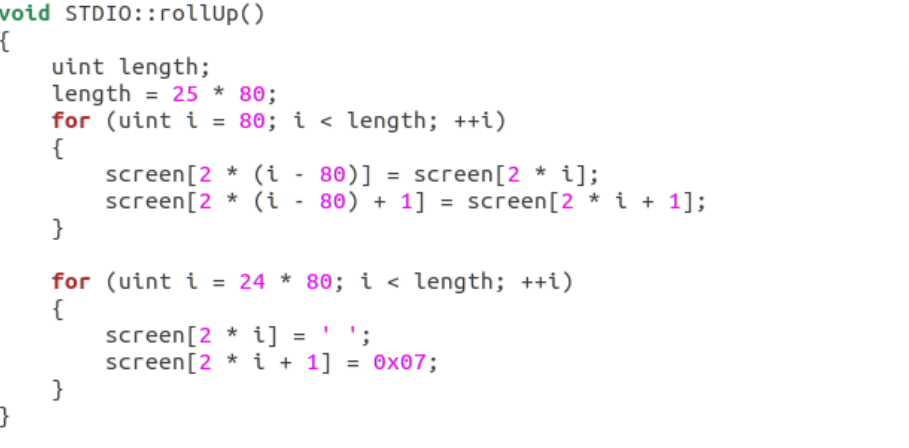
Stdio相关函数实现：

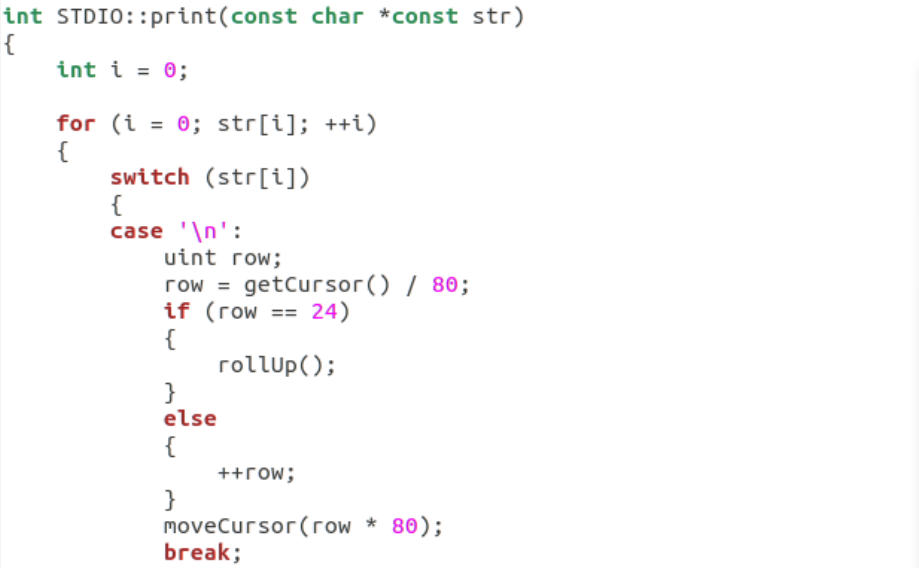


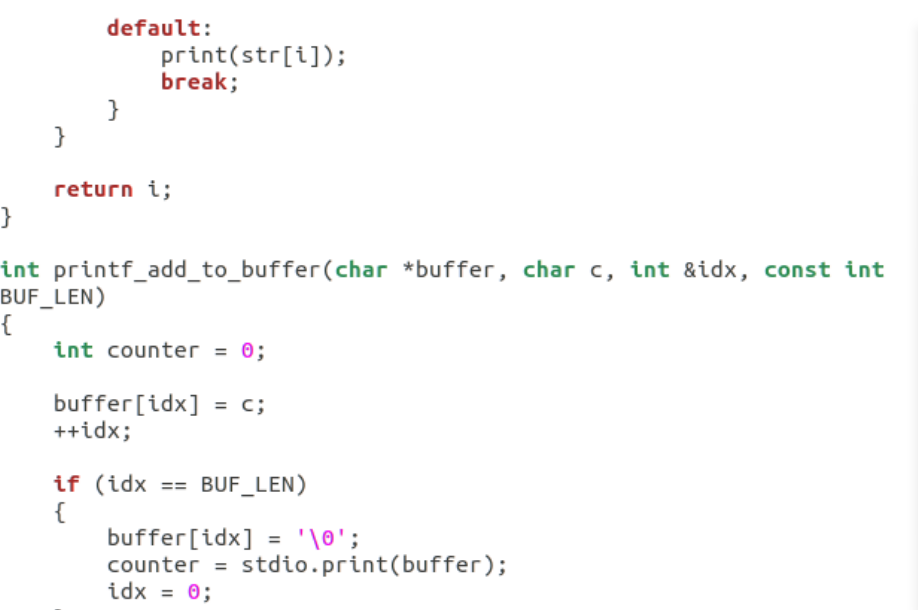


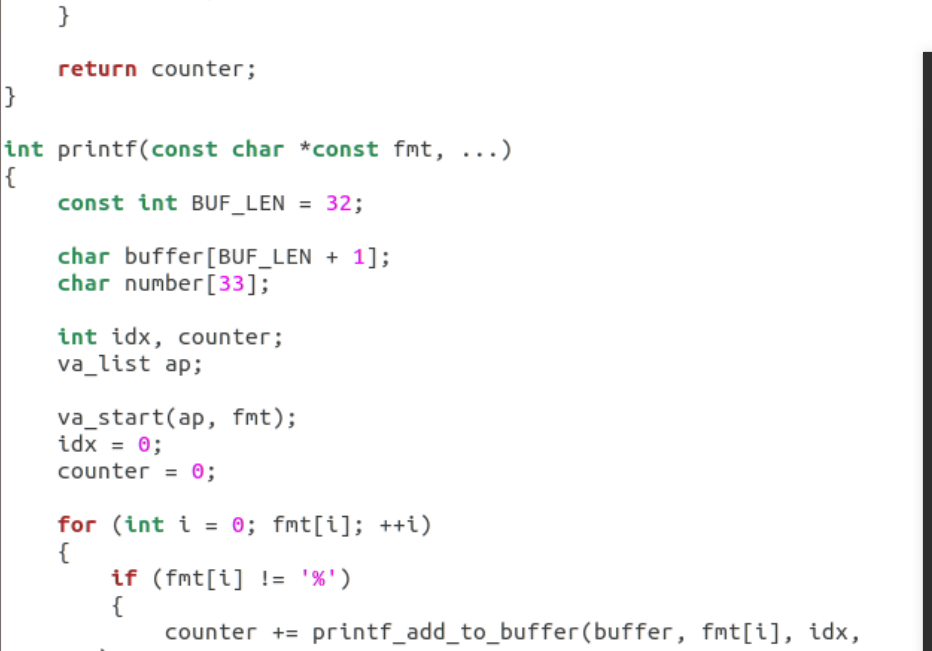


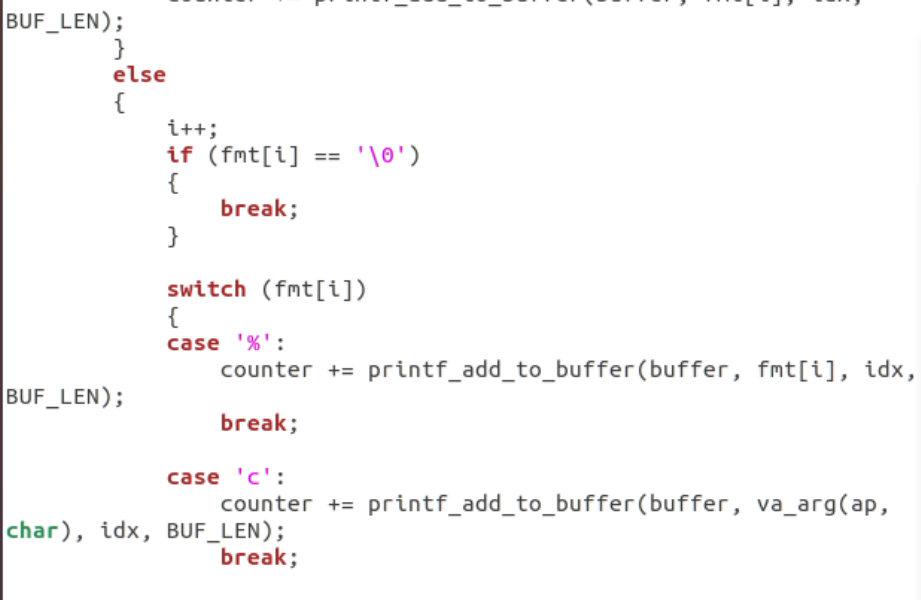


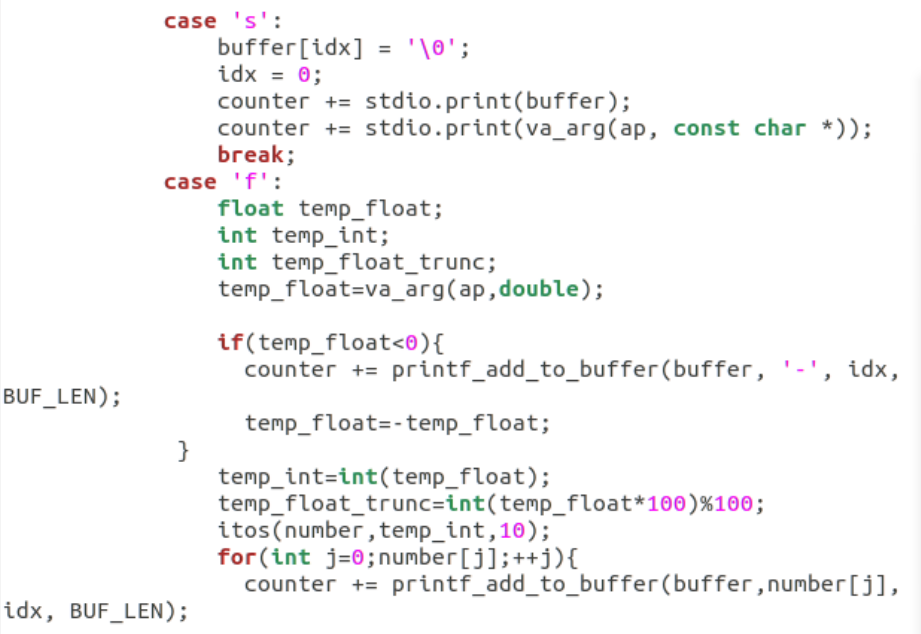


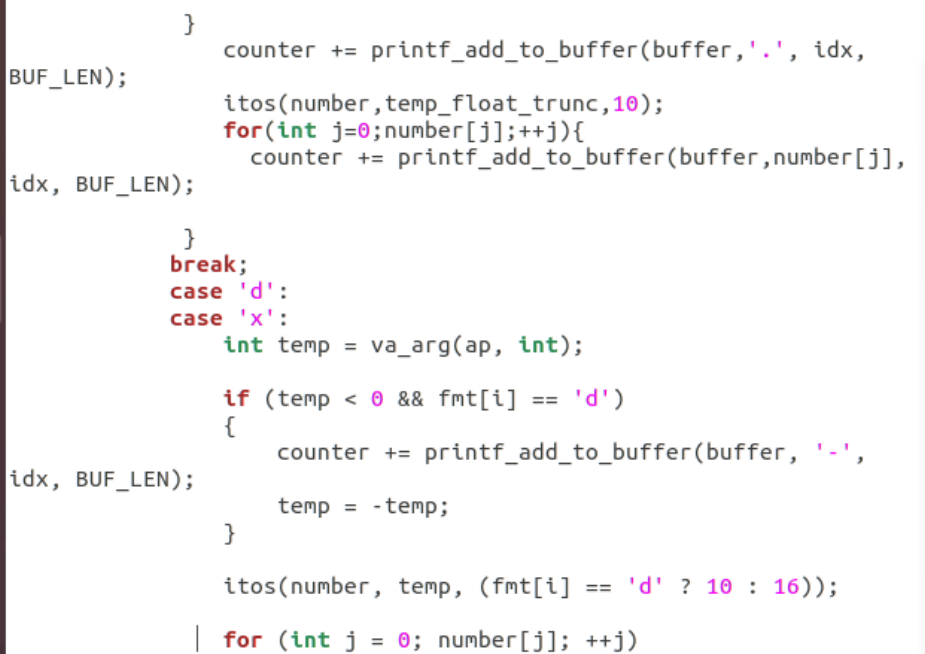


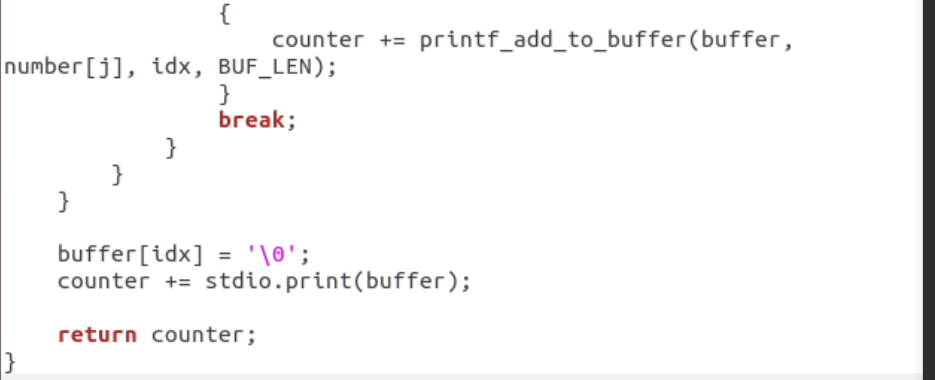




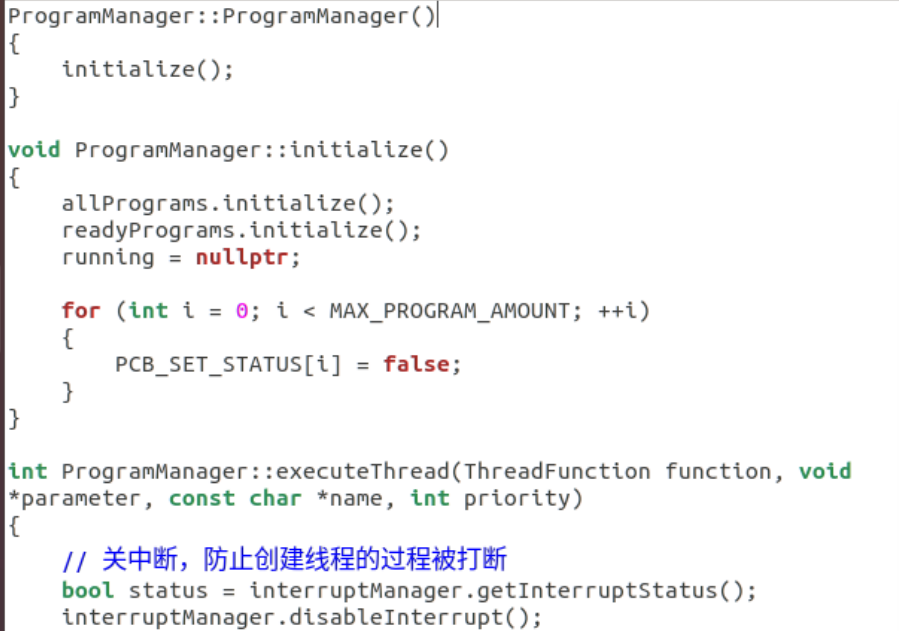


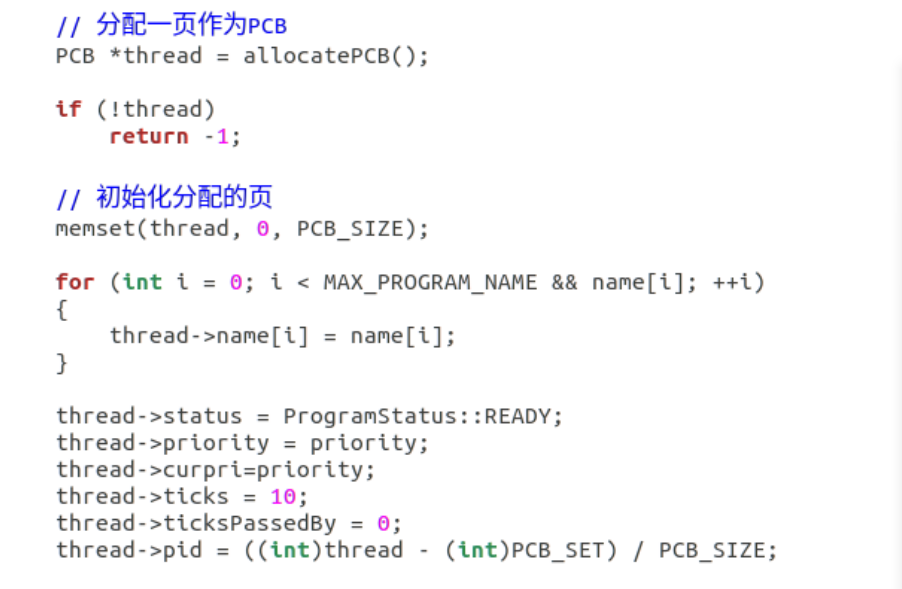


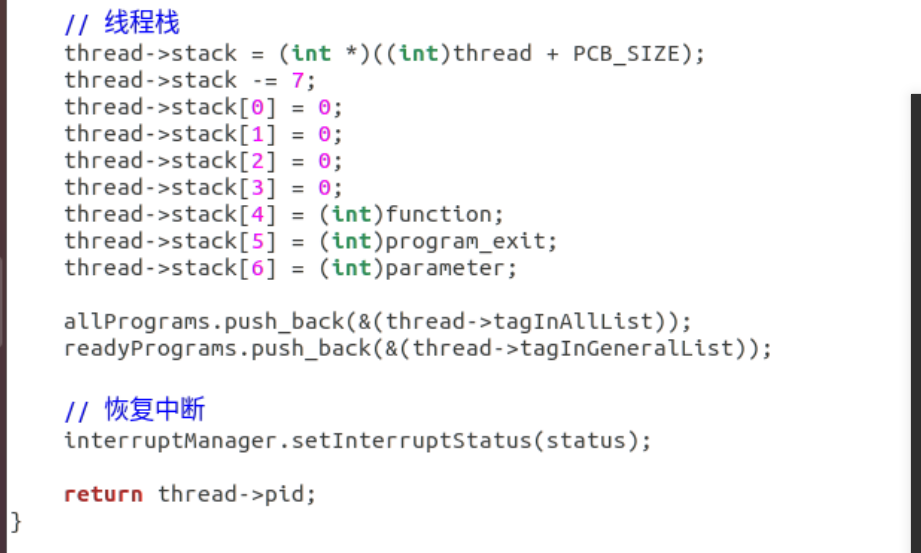


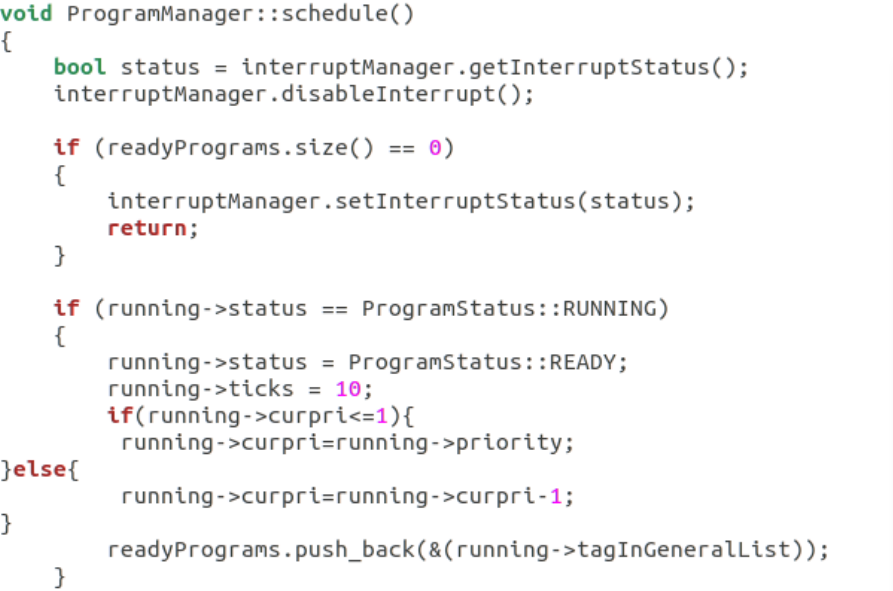
（2）Assignment 2

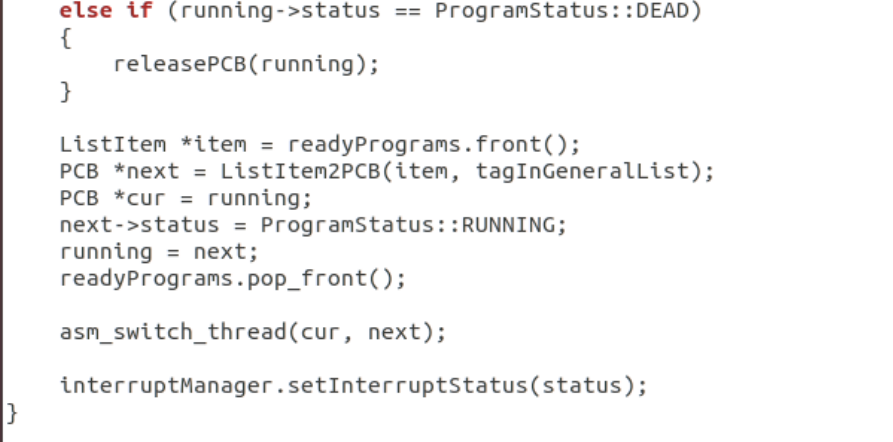
ProgramManager类中函数实现：

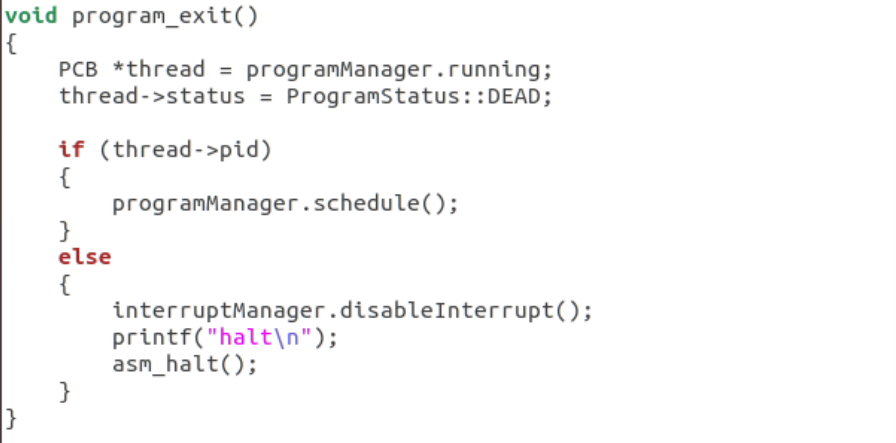


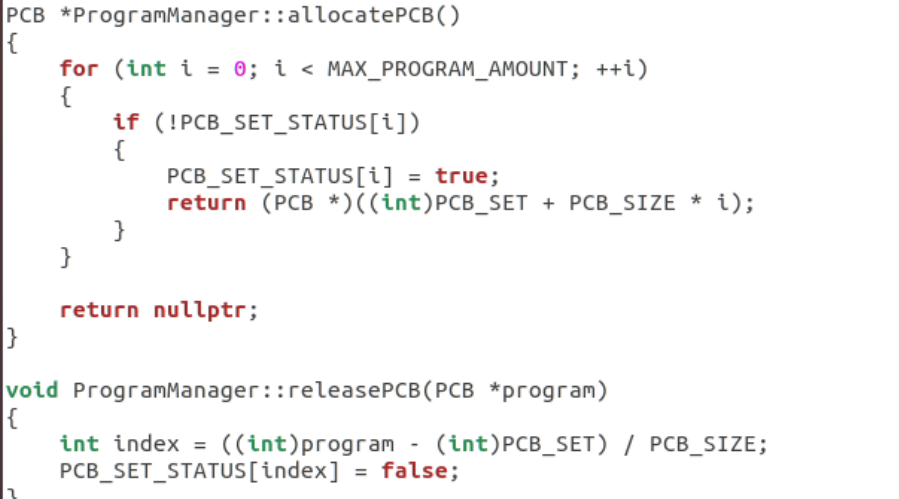




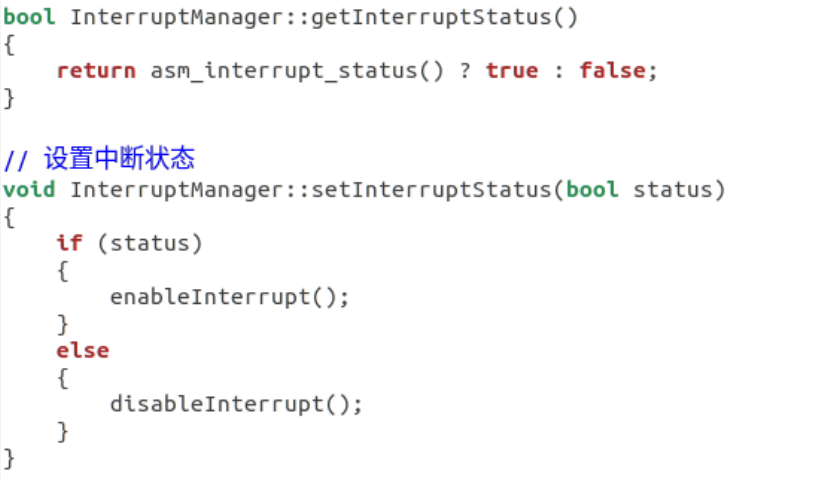












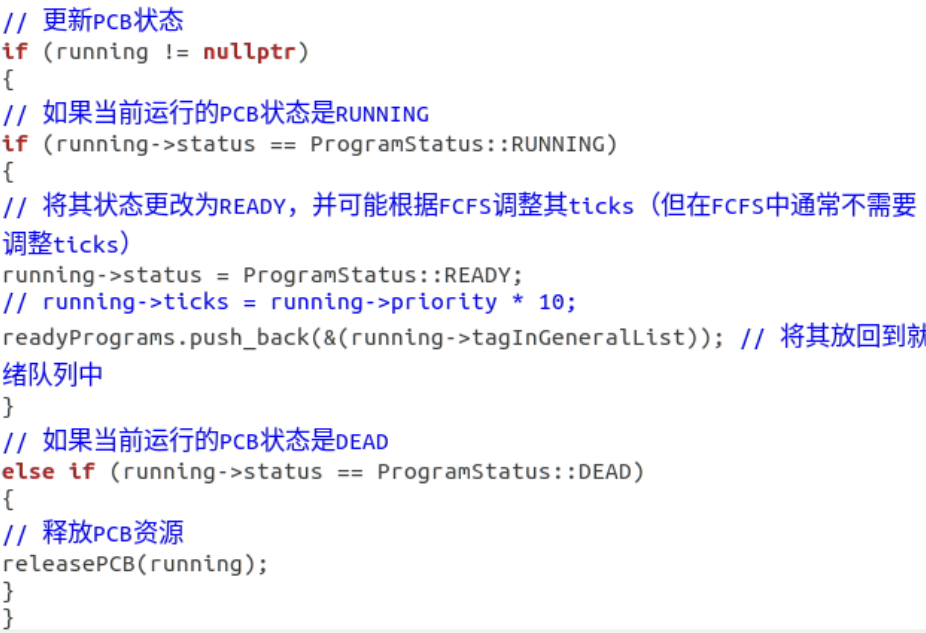
（3）Assignment 3

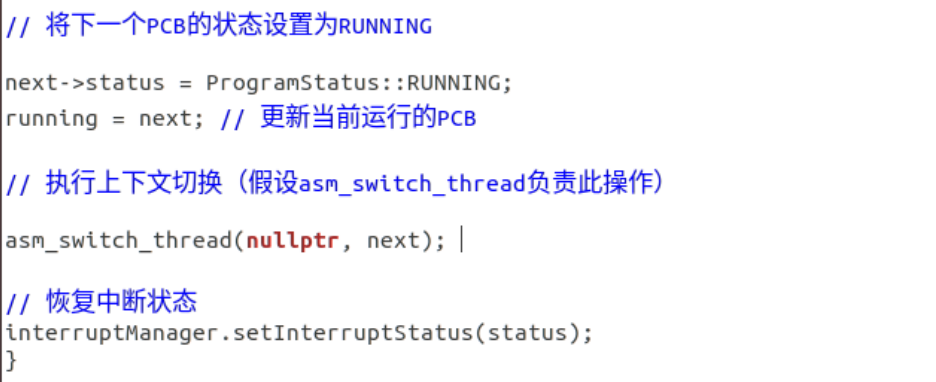
代码与示例代码保持一致，调试过程和结果已经在实验过程处说明。

（4）Assignment 4

先进先出调度算法实现：







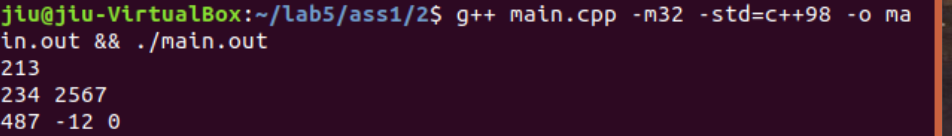
修改program\_exit函数，在某一个线程终止时，如果就绪队列非空（有线程处于 ready 状态），则进行一次调度；否则挂起：



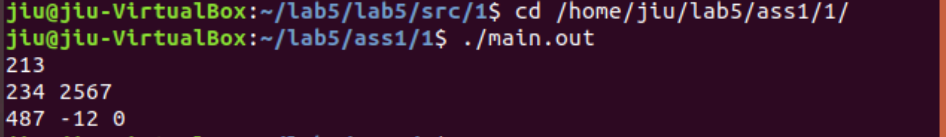
1. **实验结果**

（1）Assignment 1

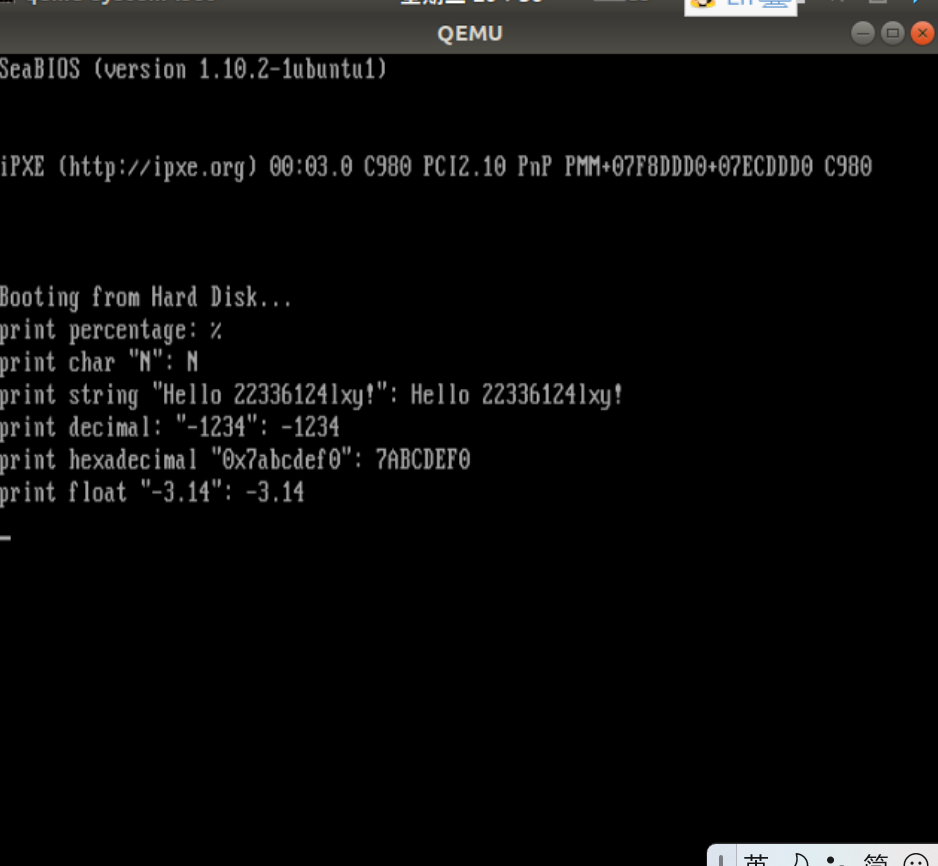
测试结果1：



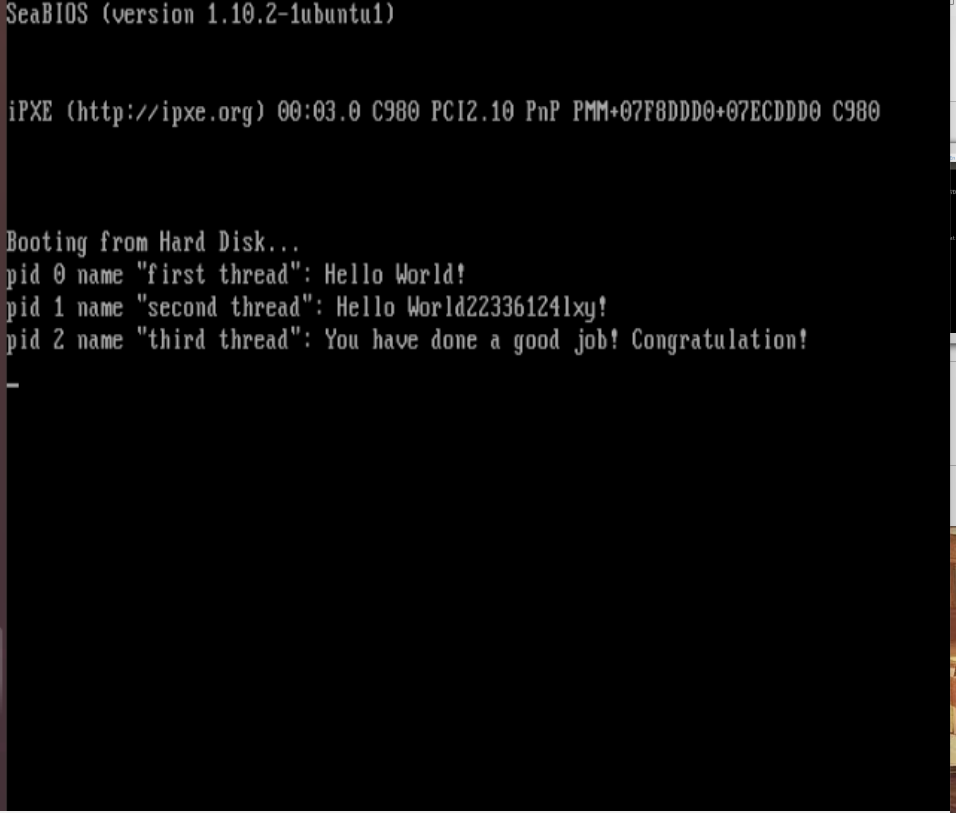
测试结果2：



Printf函数输出：



（2）Assignment 2



（3）Assignment 3

调试结果已经在实验过程处说明。

（4)Assignment 4

采用先来先服务算法，正确执行：

