1.基本信息

姓名	学号	班级	是否编译成功	运行结果是否正确
范凯涛	202221331055	计算2212	是	是

2.伪代码

```
//理发师问题

初始化

#define chair 10;        //缓冲区 (椅子数量)

int waiting;             //等待区顾客数量

sem_t baber;             //理发师信号量(是否理发)
sem_t customer;          //顾客信号量(有无顾客)
sem_t mutex;             //临界区(同时只能有一个进程改变资源数目)


//理发师函数
void baber(void)
{
    white (true) do
    {
        down(customer);                     //如无顾客,理发师睡觉(等待)
        down(mutex);                        //顾客进入临界区

        waiting = waiting - 1;                      //等候顾客数少一个

        up(baber)                   //唤醒理发师
        up(mutex)                   //释放临界区

        cut_hair();                         //开始理发
    }
}

//顾客函数
void customer(void)
{
    white (true) do
    {
        down(mutex);                    //顾客进入临界区
        if(waiting < chair)             //有空椅子
        {
            waiting++;
            up(customer);               //顾客资源加1
            up(mutex);              //释放临界区
            down(baber);                //等待理发师理发
            get_cut();                          //理发
        }

        else{
            up(mutex);                          //释放临界区
            /*
                没有椅子顾客离开
            */
        }
    }
}
```

3.主函数

头文件`shiyan5.h`

```
#ifndef shiyan5
#define shiyan5


#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>


#define NUM_THREAD 20
#define CHAIRS 10

int waiting = 0;                /*等待区顾客数量*/

sem_t barbers;                  /*理发师信号量(是否理发)*/
sem_t customers;                /*顾客信号量(有无顾客)*/
sem_t mutex;                    /*临界区(限制椅子数目变化只能有一个线程)*/

void cut_hair();
void* barber();
void get_cut();
void* customer();

#endif // shiyan5
```

---

main

```
#include "shiyan5.h"
#include "barber.c"
#include "customer.c"


int main()
{
        int retval = 0;

        sem_init(&mutex, 0, 1);
        sem_init(&customers, 0, 0);      /*没有理发师*/
        sem_init(&barbers, 0, 0);        /*没有顾客*/

        /*1个理发师，20个顾客*/
        pthread_t bar, cus[NUM_THREAD];

        /*创建理发师*/
        retval = pthread_create(&bar, NULL, barber, NULL);
        if (0 != retval)
        {
                perror("pthread_create error.");
                return -1;
        }

        /*创建顾客*/
        for (int i=0; i<NUM_THREAD; i++){
                retval = pthread_create(&cus[i], NULL, customer, &i);
                if (0 != retval){
                        printf("%d \t return value: %d\n", i, retval);
                        perror("pthread_create error.");
                        return -1;
                }
        }


        /*将理发师和顾客加入阻塞*/
        pthread_join(bar, NULL);
        for (int i=0; i<NUM_THREAD; i++)
        {
                pthread_join(cus[i], NULL);
        }

        return 0;
}
```

4.顾客函数

```
#include "shiyan5.h"
```

```
void get_cut()
{
        printf("Thread ID: %ld 准备理发\n", pthread_self());
}


void* customer()
{
        sem_wait(&mutex);           /*顾客进入临界区*/
        if(waiting < CHAIRS)        /*有空椅子*/
        {
                waiting++;
                printf("顾客进店，现在店里有 %d 个顾客等待\n", waiting);
                sem_post(&customers);    /*顾客资源加1*/
                sem_post(&mutex);       /*释放临界区*/

                sem_wait(&barbers);      /*等待理发师理发*/
                /*理发*/
                get_cut();
        }
        else
        {
                printf("没有椅子顾客离开\n");
                sem_post(&mutex);       /*释放临界区*/
        }

        pthread_exit(NULL);
        /*释放进程*/
}
```

5.理发师函数

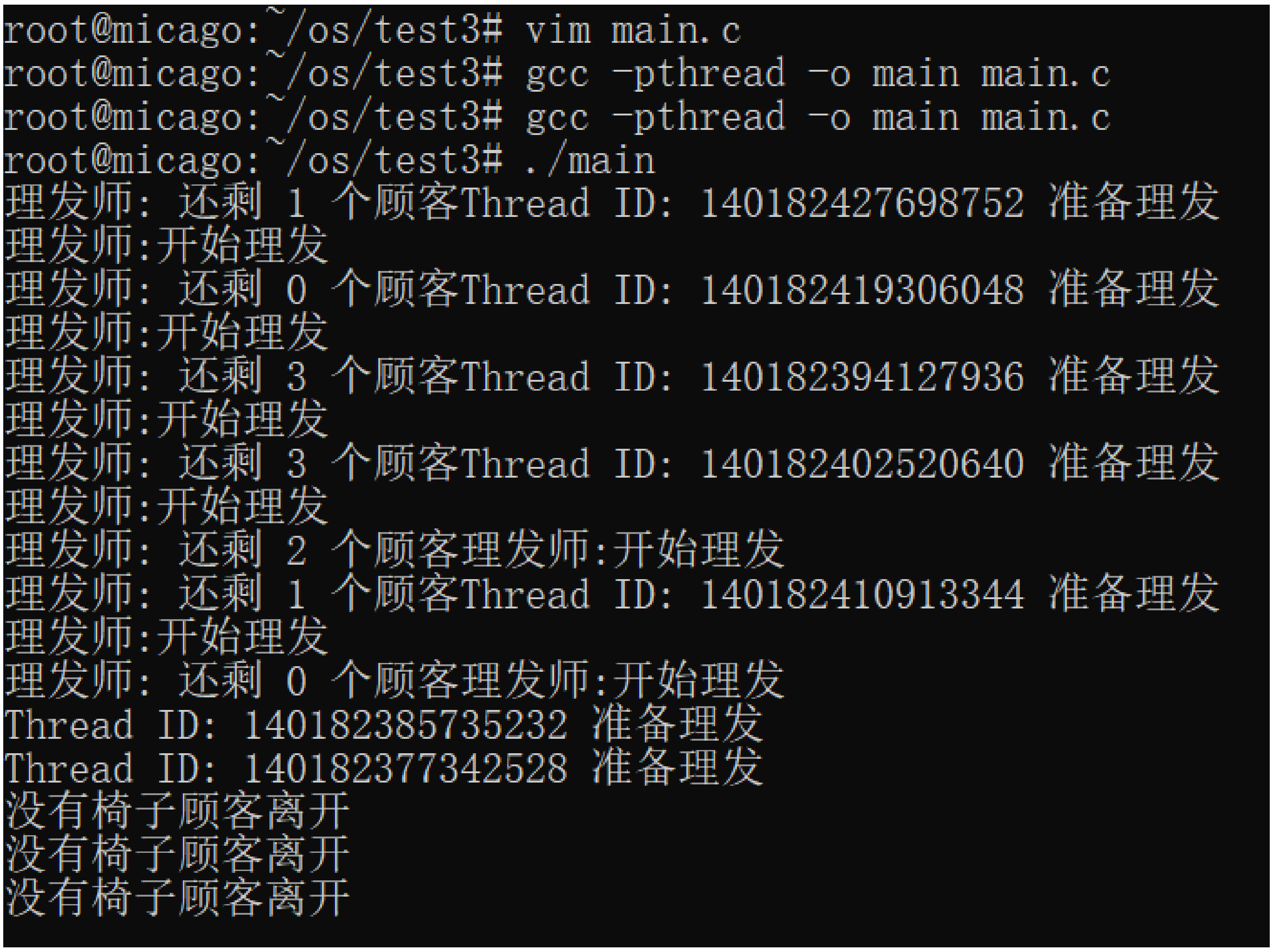
```
#include "shiyan5.h"

void cut_hair()
{
        printf("理发师:开始理发\n");
        sleep(1);
}

/*理发师函数*/

void* barber()
{
        while(1)
        {
                sem_wait(&customers);                                   /*如无顾
客,理发师睡觉(等待)*/
                sem_wait(&mutex);                                           /*
顾客进入临界区*/
```

```
                waiting = waiting - 1;                                  /*等候顾客
数少一个*/
                printf("理发师: 还剩 %d 个顾客\n", waiting);

                sem_post(&barbers);                                             /*
唤醒理发师*/
                sem_post(&mutex);                                               /*
释放临界区*/
                cut_hair();
                /*开始理发*/

                sem_wait(&mutex);
                if(waiting == 0)
                {
                        sem_post(&mutex);
                        break;
                }
                sem_post(&mutex);
        }
        printf("没有顾客，理发师去睡觉了\n");
        pthread_exit(NULL);
}
```

6.运行结果及分析

第一次运行

运行结果



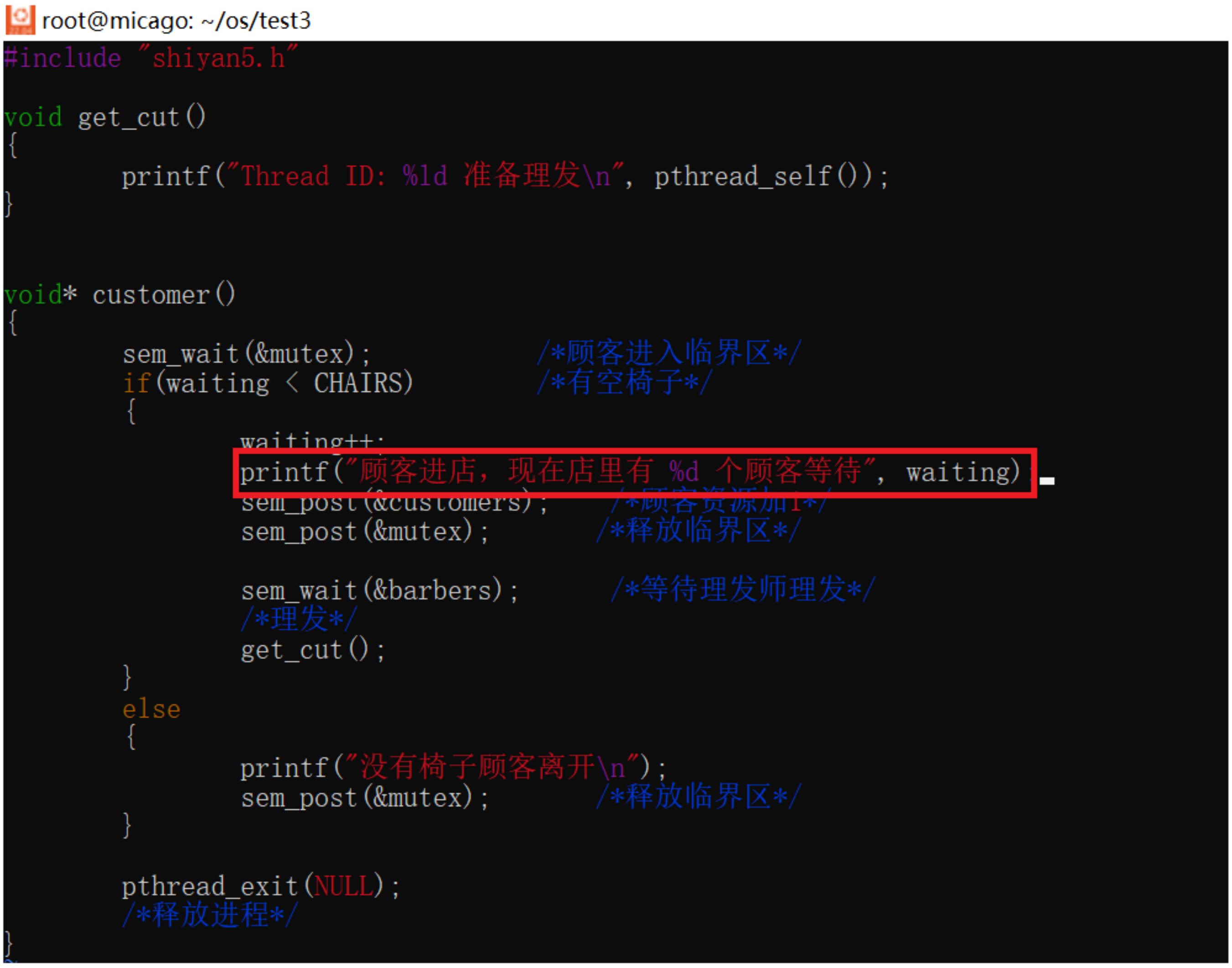
```
root@micago:~/os/test3# vim main.c
root@micago:~/os/test3# gcc -pthread -o main main.c
root@micago:~/os/test3# gcc -pthread -o main main.c
root@micago:~/os/test3# ./main
理发师: 还剩 1 个顾客Thread ID: 140182427698752 准备理发
理发师:开始理发
理发师: 还剩 0 个顾客Thread ID: 140182419306048 准备理发
理发师:开始理发
理发师: 还剩 3 个顾客Thread ID: 140182394127936 准备理发
理发师:开始理发
理发师: 还剩 3 个顾客Thread ID: 140182402520640 准备理发
理发师:开始理发
理发师: 还剩 2 个顾客理发师:开始理发
理发师: 还剩 1 个顾客Thread ID: 140182410913344 准备理发
理发师:开始理发
理发师: 还剩 0 个顾客理发师:开始理发
Thread ID: 140182385735232 准备理发
Thread ID: 140182377342528 准备理发
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
```

分析

我定义了1个理发师及20个顾客，但并没有完整运行，只有10个顾客进入了理发店，疑似在运行途中遇到了死锁导致了程序卡死。

处理

在顾客进店时加入提示信息，检查死锁产生代码

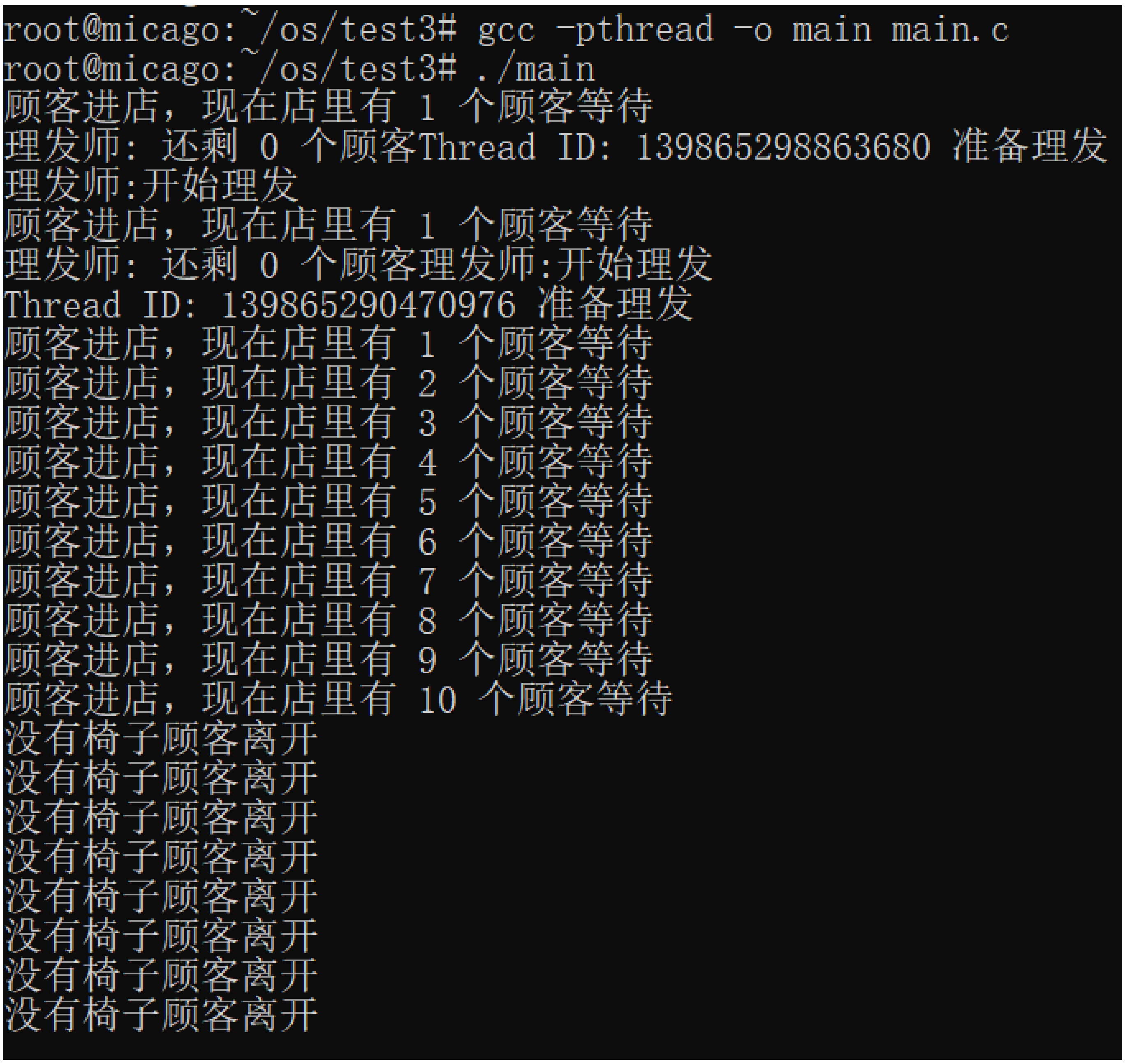


root@micago: ~/os/test3

```
#include "shiyan5.h"

void get_cut()
{
        printf("Thread ID: %ld 准备理发\n", pthread_self());
}


void* customer()
{
        sem_wait(&mutex);               /*顾客进入临界区*/
        if(waiting < CHAIRS)            /*有空椅子*/
        {
                waiting++;
                printf("顾客进店，现在店里有 %d 个顾客等待", waiting);
                sem_post(&customers);       /*顾客资源加1*/
                sem_post(&mutex);          /*释放临界区*/

                sem_wait(&barbers);         /*等待理发师理发*/
                /*理发*/
                get_cut();
        }
        else
        {
                printf("没有椅子顾客离开\n");
                sem_post(&mutex);          /*释放临界区*/
        }

        pthread_exit(NULL);
        /*释放进程*/
}
```

第二次测试:

运行结果



```
root@micago:~/os/test3# gcc -pthread -o main main.c
root@micago:~/os/test3# ./main
顾客进店，现在店里有 1 个顾客等待
理发师: 还剩 0 个顾客Thread ID: 139865298863680 准备理发
理发师:开始理发
顾客进店，现在店里有 1 个顾客等待
理发师: 还剩 0 个顾客理发师:开始理发
Thread ID: 139865290470976 准备理发
顾客进店，现在店里有 1 个顾客等待
顾客进店，现在店里有 2 个顾客等待
顾客进店，现在店里有 3 个顾客等待
顾客进店，现在店里有 4 个顾客等待
顾客进店，现在店里有 5 个顾客等待
顾客进店，现在店里有 6 个顾客等待
顾客进店，现在店里有 7 个顾客等待
顾客进店，现在店里有 8 个顾客等待
顾客进店，现在店里有 9 个顾客等待
顾客进店，现在店里有 10 个顾客等待
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
```

分析

发现20顾客进店是正常的,而理发师只理了两个顾客就停止了，故判断死锁产生在理发师进行理发时

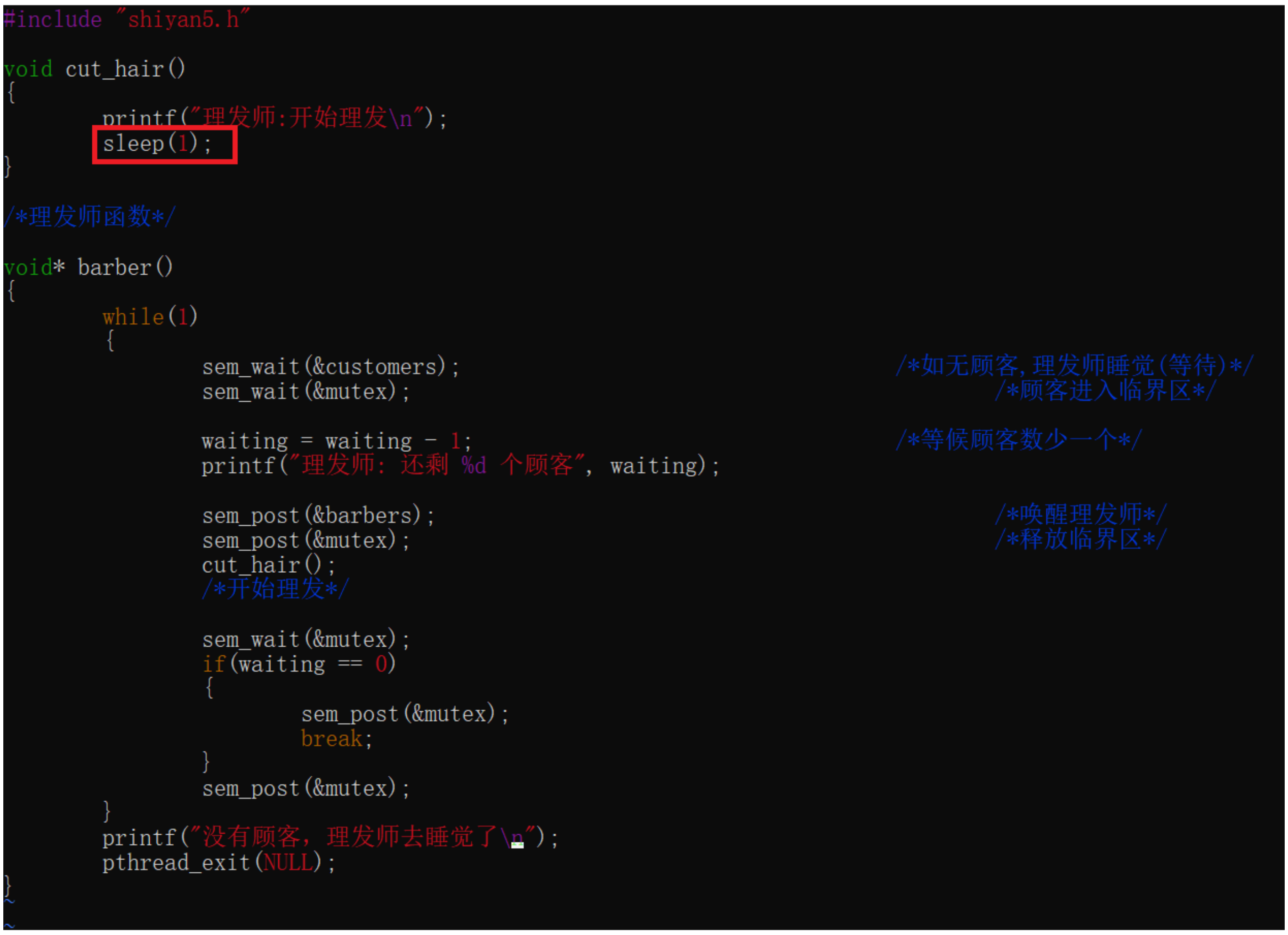
处理

检查理发师代码发现，理发函数没有加上延迟，顾客一进店就瞬间完成了理发，导致后续顾客来不及进店，理发师判断无顾客直接睡觉去了，终止了进程，后续的顾客应为缺少理发师而卡死在等待理发的步骤。

第三次测试

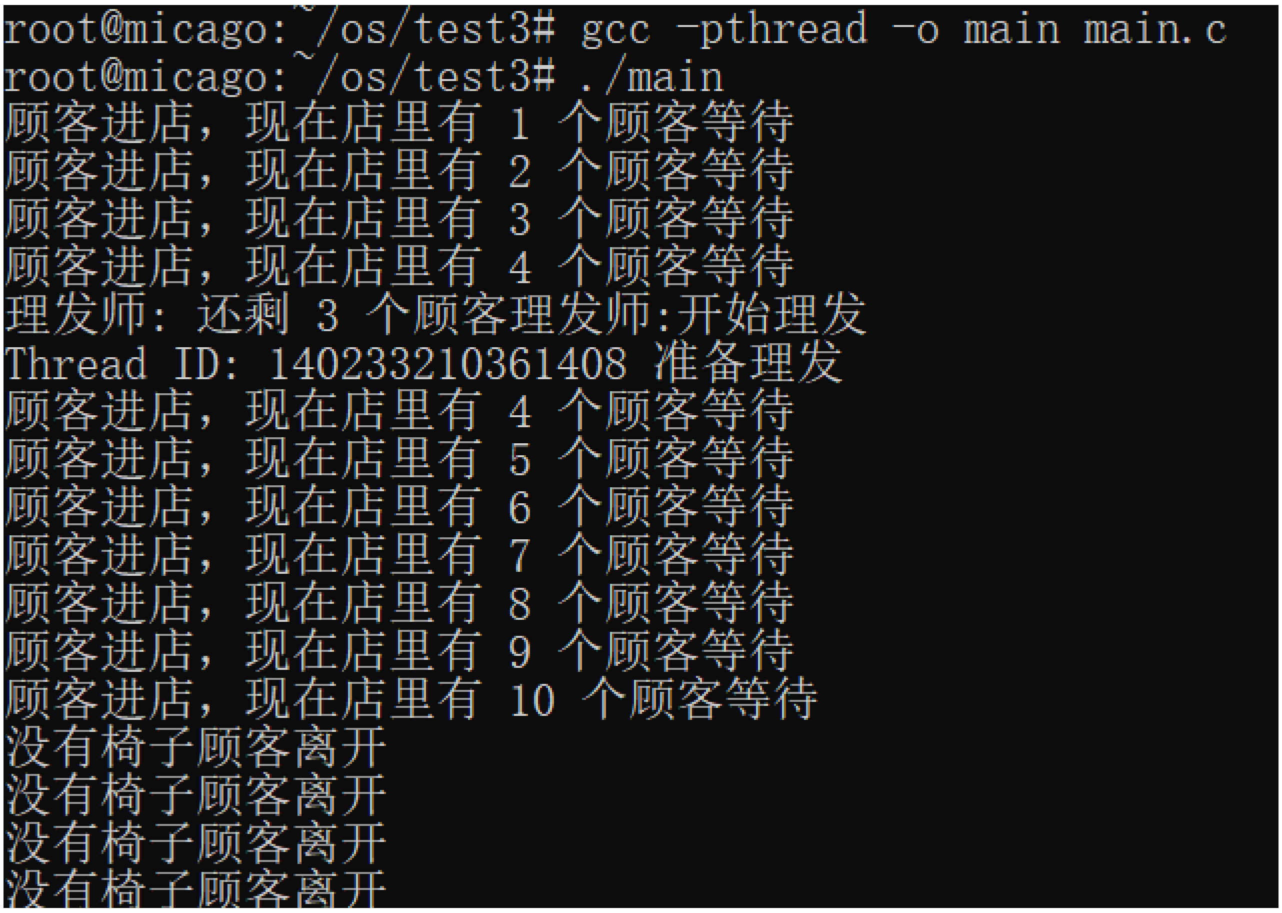
处理

在理发师函数的理发函数中加上理发的延时

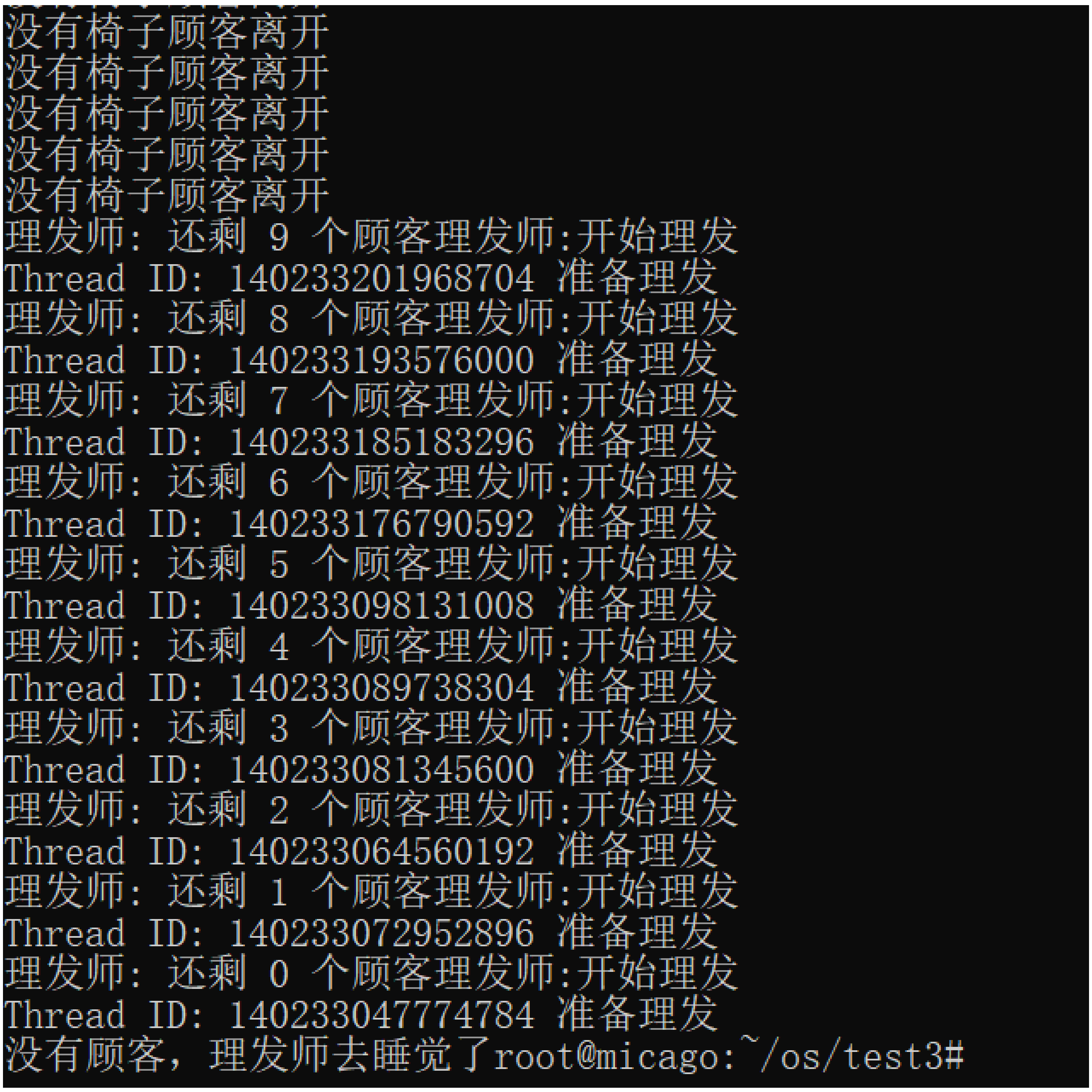


```
#include "shiyan5.h"

void cut_hair()
{
        printf("理发师:开始理发\n");
        sleep(1);
}

/*理发师函数*/

void* barber()
{
        while(1)
        {
                sem_wait(&customers);                                          /*如无顾客,理发师睡觉(等待)*/
                sem_wait(&mutex);                                                      /*顾客进入临界区*/

                waiting = waiting - 1;                                         /*等候顾客数少一个*/
                printf("理发师: 还剩 %d 个顾客", waiting);

                sem_post(&barbers);                                                    /*唤醒理发师*/
                sem_post(&mutex);                                                      /*释放临界区*/
                cut_hair();
                /*开始理发*/

                sem_wait(&mutex);
                if(waiting == 0)
                {
                        sem_post(&mutex);
                        break;
                }
                sem_post(&mutex);
        }
        printf("没有顾客, 理发师去睡觉了\n");
        pthread_exit(NULL);
}
~
~
```

运行结果



```
root@micago:~/os/test3# gcc -pthread -o main main.c
root@micago:~/os/test3# ./main
顾客进店, 现在店里有 1 个顾客等待
顾客进店, 现在店里有 2 个顾客等待
顾客进店, 现在店里有 3 个顾客等待
顾客进店, 现在店里有 4 个顾客等待
理发师: 还剩 3 个顾客理发师:开始理发
Thread ID: 140233210361408 准备理发
顾客进店, 现在店里有 4 个顾客等待
顾客进店, 现在店里有 5 个顾客等待
顾客进店, 现在店里有 6 个顾客等待
顾客进店, 现在店里有 7 个顾客等待
顾客进店, 现在店里有 8 个顾客等待
顾客进店, 现在店里有 9 个顾客等待
顾客进店, 现在店里有 10 个顾客等待
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
```



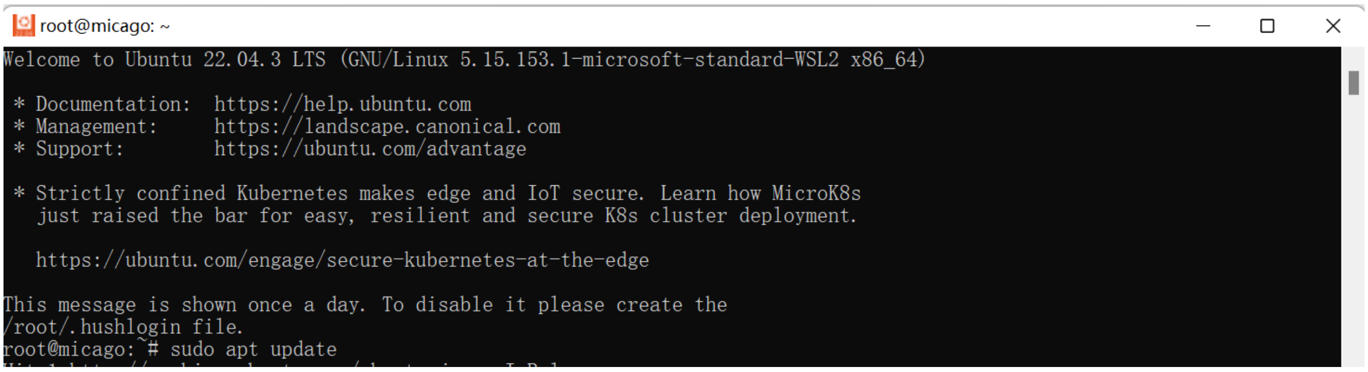
```
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
没有椅子顾客离开
理发师: 还剩 9 个顾客理发师:开始理发
Thread ID: 140233201968704 准备理发
理发师: 还剩 8 个顾客理发师:开始理发
Thread ID: 140233193576000 准备理发
理发师: 还剩 7 个顾客理发师:开始理发
Thread ID: 140233185183296 准备理发
理发师: 还剩 6 个顾客理发师:开始理发
Thread ID: 140233176790592 准备理发
理发师: 还剩 5 个顾客理发师:开始理发
Thread ID: 140233098131008 准备理发
理发师: 还剩 4 个顾客理发师:开始理发
Thread ID: 140233089738304 准备理发
理发师: 还剩 3 个顾客理发师:开始理发
Thread ID: 140233081345600 准备理发
理发师: 还剩 2 个顾客理发师:开始理发
Thread ID: 140233064560192 准备理发
理发师: 还剩 1 个顾客理发师:开始理发
Thread ID: 140233072952896 准备理发
理发师: 还剩 0 个顾客理发师:开始理发
Thread ID: 140233047774784 准备理发
没有顾客，理发师去睡觉了root@micago:~/os/test3#
```

成功解决问题，完成实验理发师问题！

7.实验中遇到的问题及解决办法

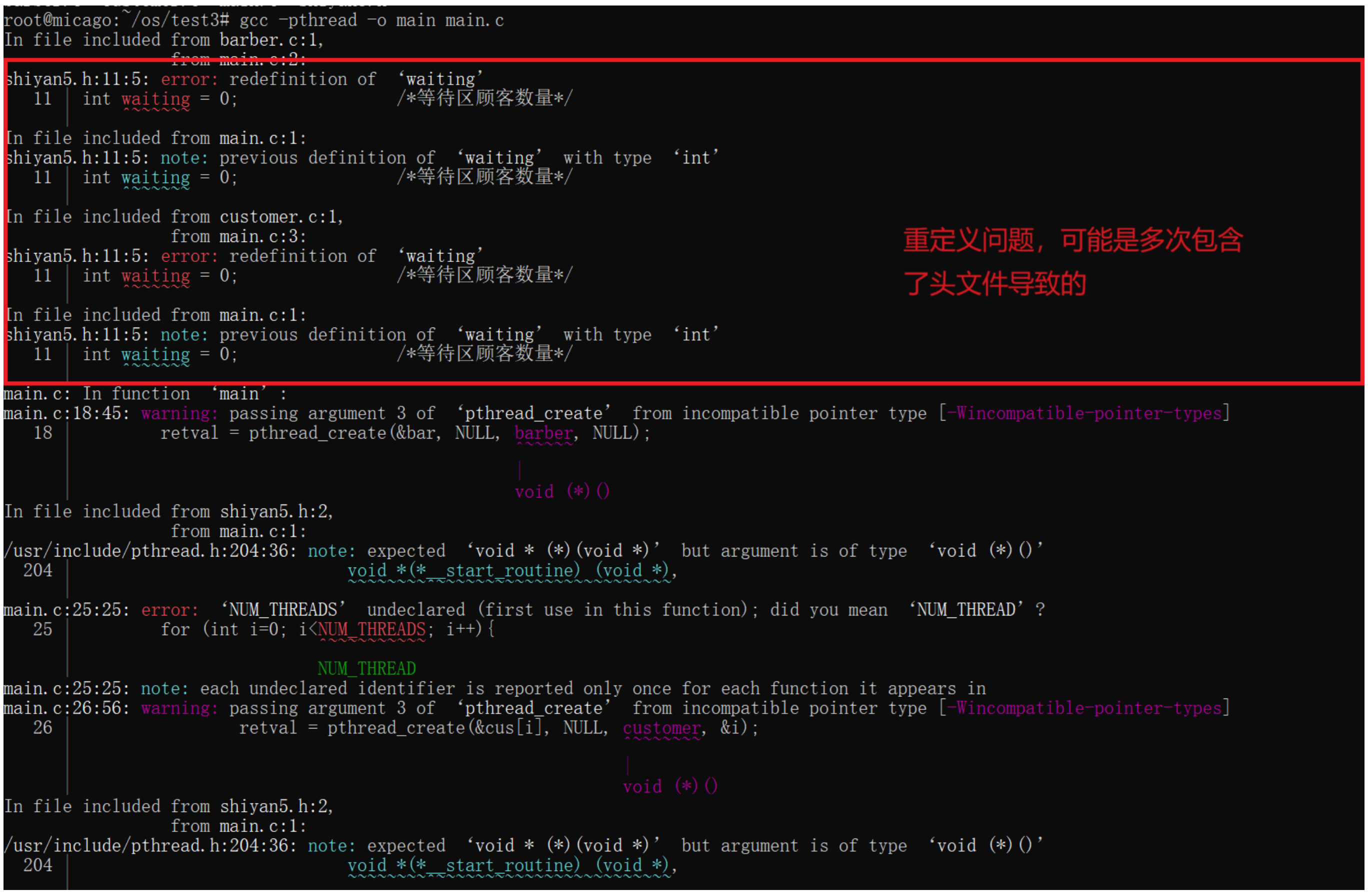
1. 电脑上没有linux环境，不方便调试代码

解决：安装ubuntu并安装gcc用于模拟linux环境调试代码



```
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/root/.hushlogin file.
root@micago:~# sudo apt update
```

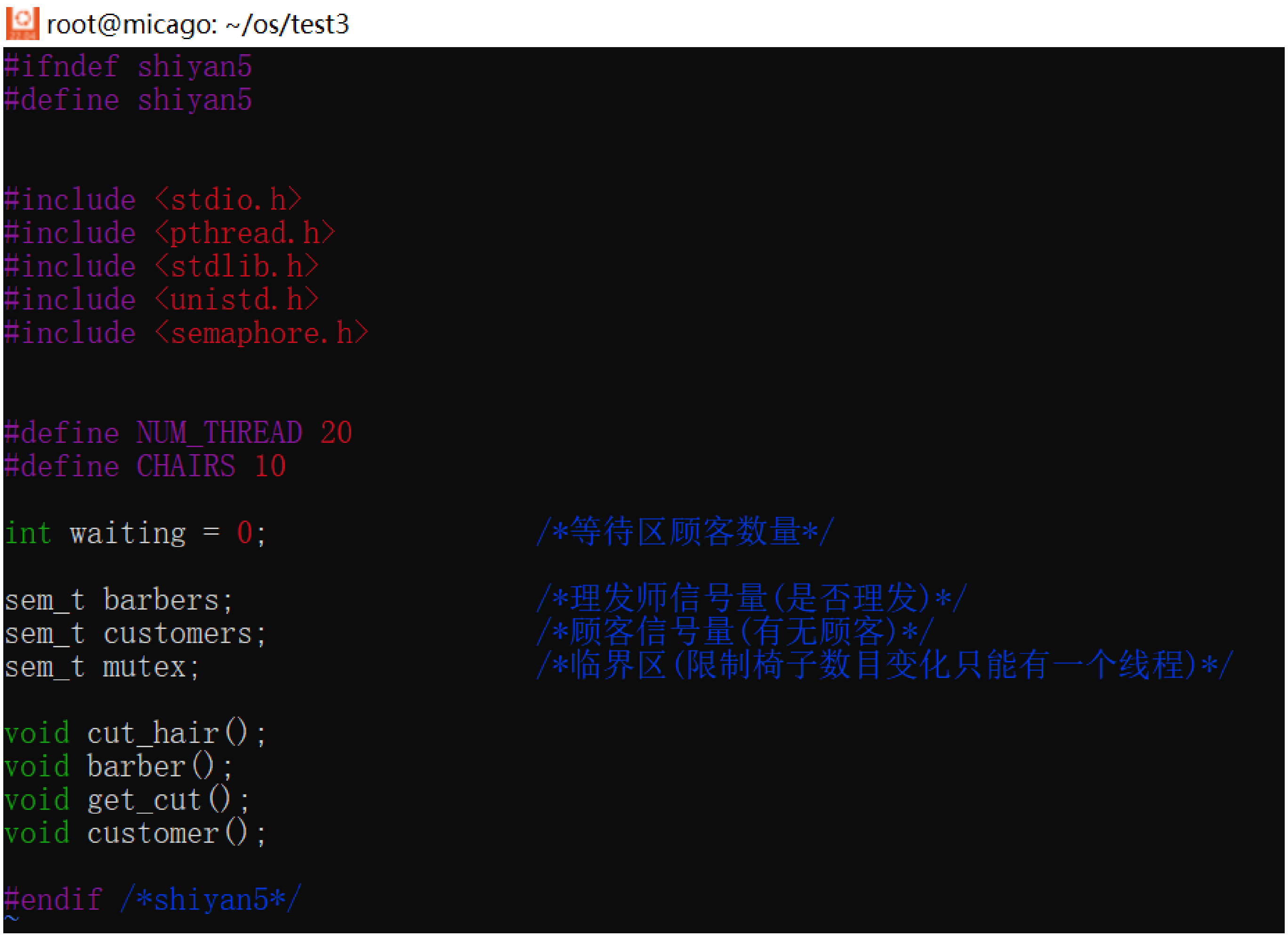
2. 重定义问题



```
root@micago:~/os/test3# gcc -pthread -o main main.c
In file included from barber.c:1,
                 from main.c:2:
shiyan5.h:11:5: error: redefinition of ‘waiting’
   11 | int waiting = 0;                /*等待区顾客数量*/
In file included from main.c:1:
shiyan5.h:11:5: note: previous definition of ‘waiting’ with type ‘int’
   11 | int waiting = 0;                /*等待区顾客数量*/
In file included from customer.c:1,
                 from main.c:3:
shiyan5.h:11:5: error: redefinition of ‘waiting’
   11 | int waiting = 0;                /*等待区顾客数量*/
In file included from main.c:1:
shiyan5.h:11:5: note: previous definition of ‘waiting’ with type ‘int’
   11 | int waiting = 0;                /*等待区顾客数量*/
main.c: In function ‘main’:
main.c:18:45: warning: passing argument 3 of ‘pthread_create’ from incompatible pointer type [-Wincompatible-pointer-types]
   18 |         retval = pthread_create(&bar, NULL, barber, NULL);
      |                                             |
      |                                             void (*)()
In file included from shiyan5.h:2,
                 from main.c:1:
/usr/include/pthread.h:204:36: note: expected ‘void * (*)(void *)’ but argument is of type ‘void (*)()’
  204 |                            void *(*__start_routine) (void *),
main.c:25:25: error: ‘NUM_THREADS’ undeclared (first use in this function); did you mean ‘NUM_THREAD’?
   25 |         for (int i=0; i<NUM_THREADS; i++){
      |                         NUM_THREAD
main.c:25:25: note: each undeclared identifier is reported only once for each function it appears in
main.c:26:56: warning: passing argument 3 of ‘pthread_create’ from incompatible pointer type [-Wincompatible-pointer-types]
   26 |                 retval = pthread_create(&cus[i], NULL, customer, &i);
      |                                                        |
      |                                                        void (*)()
In file included from shiyan5.h:2,
                 from main.c:1:
/usr/include/pthread.h:204:36: note: expected ‘void * (*)(void *)’ but argument is of type ‘void (*)()’
  204 |                            void *(*__start_routine) (void *),
```

重定义问题，可能是多次包含了头文件导致的

解决:在头文件中加入ifndef 来限制只包含一次头文件



root@micago: ~/os/test3

```
#ifndef shiyan5
#define shiyan5


#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>


#define NUM_THREAD 20
#define CHAIRS 10

int waiting = 0;                /*等待区顾客数量*/

sem_t barbers;                  /*理发师信号量(是否理发)*/
sem_t customers;                /*顾客信号量(有无顾客)*/
sem_t mutex;                    /*临界区(限制椅子数目变化只能有一个线程)*/

void cut_hair();
void barber();
void get_cut();
void customer();

#endif /*shiyan5*/
~
```

3. 函数参数不对

pthread_create 函数传参传的是函数指针

2. 线程创建pthread_create

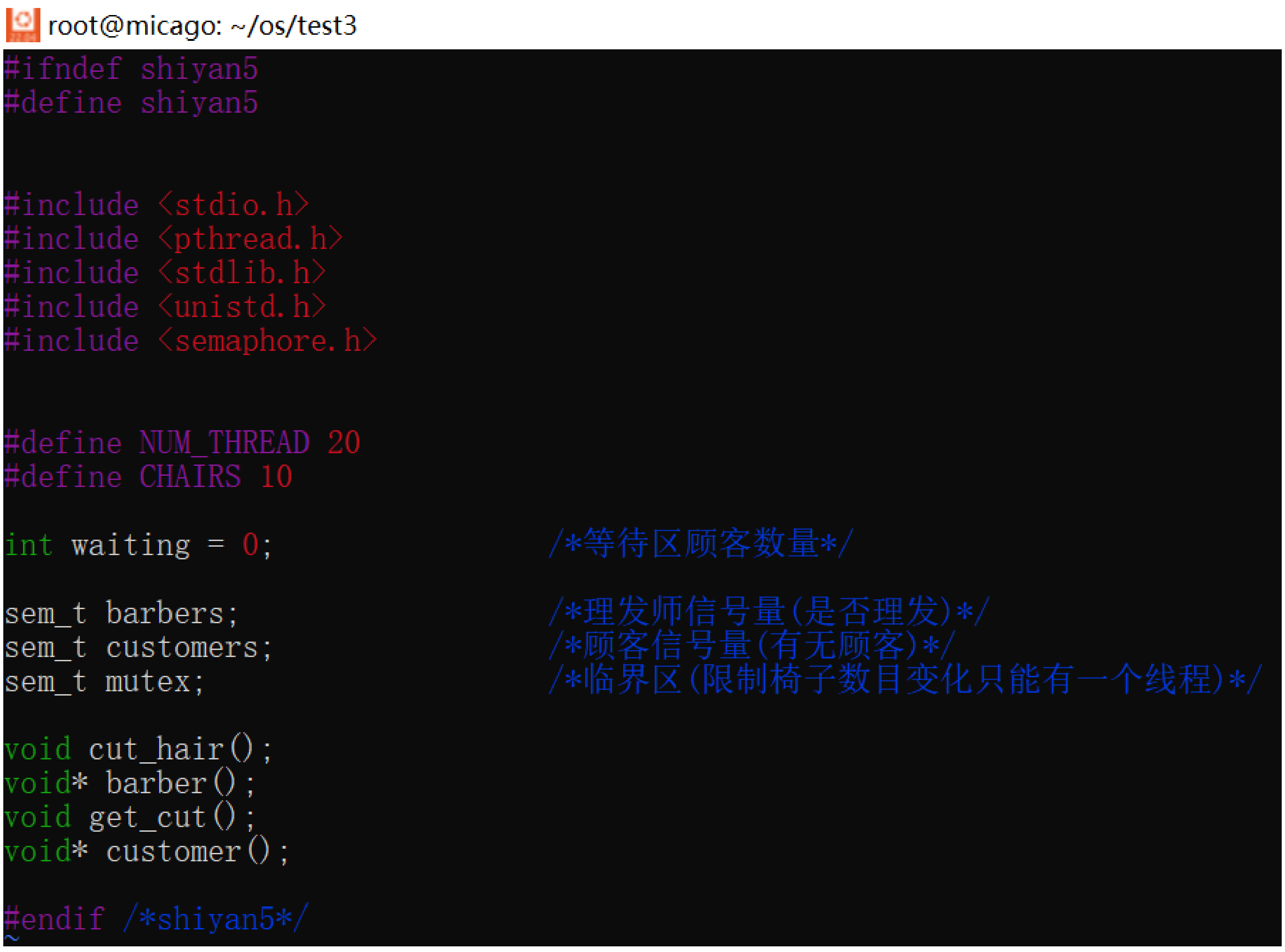
线程创建函数：pthread_create(3) - Linux manual page

```
#include <pthread.h>

int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine) (void *),
                   void *arg);
```

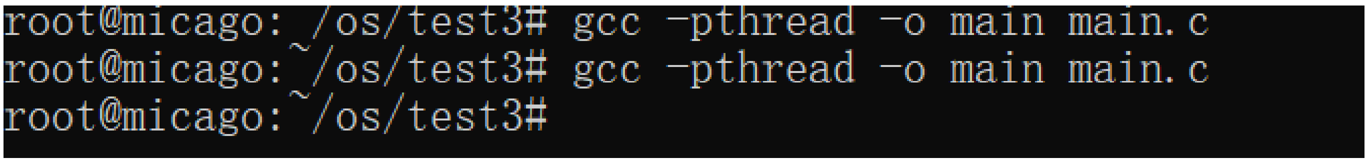
- `pthread_t *thread`，线程创建后返回的线程ID，类型实为 `unsigned long int`
- `void *(*start_routine) (void *)`，是一个函数指数，线程执行的代码
- `void *arg`，传给线程函数 `start_routine` 的参数，只能一个

`pthread_create` 创建的线程默认是joinable，可以调用 `pthread_join` 函数。

解决: 修改头文件和对应函数参数


```
root@micago: ~/os/test3
#ifndef shiyan5
#define shiyan5


#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>


#define NUM_THREAD 20
#define CHAIRS 10

int waiting = 0;                /*等待区顾客数量*/

sem_t barbers;                  /*理发师信号量(是否理发)*/
sem_t customers;                /*顾客信号量(有无顾客)*/
sem_t mutex;                    /*临界区(限制椅子数目变化只能有一个线程)*/

void cut_hair();
void* barber();
void get_cut();
void* customer();

#endif /*shiyan5*/
~
```

解决所有问题成功编译


```
root@micago:~/os/test3# gcc -pthread -o main main.c
root@micago:~/os/test3# gcc -pthread -o main main.c
root@micago:~/os/test3#
```

8.附完整源代码

```
#ifndef shiyan5
#define shiyan5


#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>


#define NUM_THREAD 20
#define CHAIRS 10
```

```
int waiting = 0;                /*等待区顾客数量*/

sem_t barbers;                  /*理发师信号量(是否理发)*/
sem_t customers;                /*顾客信号量(有无顾客)*/
sem_t mutex;                    /*临界区(限制椅子数目变化只能有一个线程)*/

void cut_hair();
void* barber();
void get_cut();
void* customer();

#endif // shiyan5

--------------------
#include "shiyan5.h"

void cut_hair()
{
        printf("理发师:开始理发\n");
        sleep(1);
}

/*理发师函数*/

void* barber()
{
        while(1)
        {
                sem_wait(&customers);                                   /*如无顾
客,理发师睡觉(等待)*/
                sem_wait(&mutex);                                               /*
顾客进入临界区*/

                waiting = waiting - 1;                                  /*等候顾客
数少一个*/
                printf("理发师: 还剩 %d 个顾客\n", waiting);

                sem_post(&barbers);                                             /*
唤醒理发师*/
                sem_post(&mutex);                                               /*
释放临界区*/
                cut_hair();
                /*开始理发*/

                sem_wait(&mutex);
                if(waiting == 0)
                {
                        sem_post(&mutex);
                        break;
                }
                sem_post(&mutex);
        }
        printf("没有顾客, 理发师去睡觉了\n");
        pthread_exit(NULL);
}

---------------------
#include "shiyan5.h"

void get_cut()
{
printf("Thread ID: %ld 准备理发\n", pthread_self());
}


void customer()
{
sem_wait(&mutex);            /*顾客进入临界区*/
if(waiting < CHAIRS)         /*有空椅子*/
{
waiting++;
sem_post(&customers);    /*顾客资源加1*/
sem_post(&mutex);       /*释放临界区*/

sem_wait(&barbers);      /*等待理发师理发*/
/*理发*/
get_cut();
}
else
{
printf("没有椅子顾客离开\n");
sem_post(&mutex);       /*释放临界区*/
}

pthread_exit(NULL);
/*释放进程*/
}

--------------------
#include "shiyan5.h"
#include "barber.c"
#include "customer.c"


int main()
{
    int retval = 0;
    sem_init(&mutex, 0, 1);
    sem_init(&customers, 0, 0);      /*没有理发师*/
    sem_init(&barbers, 0, 0);        /*没有顾客*/
    /*1个理发师，20个顾客*/
    pthread_t bar, cus[NUM_THREAD];
    /*创建理发师*/
    retval = pthread_create(&bar, NULL, barber, NULL);
    if (0 != retval)
    {
            perror("pthread_create error.");
            return -1;
    }
    /*创建顾客*/
    for (int i=0; i<NUM_THREAD; i++){
            retval = pthread_create(&cus[i], NULL, customer, &i);
            if (0 != retval){
                    printf("%d \t return value: %d\n", i, retval);
                    perror("pthread_create error.");
                    return -1;
            }
    }
    /*将理发师和顾客加入阻塞*/
    pthread_join(bar, NULL);
    for (int i=0; i<NUM_THREAD; i++)
    {
            pthread_join(cus[i], NULL);
    }
    return 0;
}
```