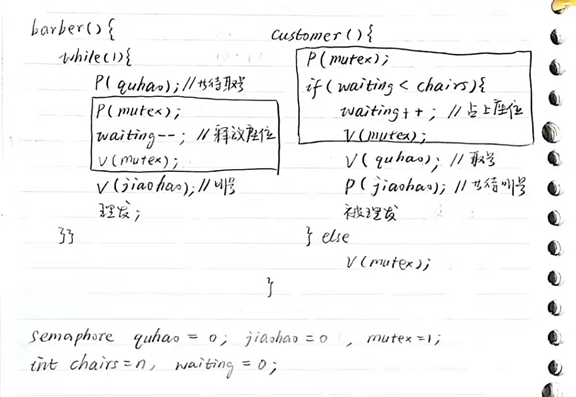
### 10. 实现"理发师问题"

通过对之前9个例程的学习, 编写一个解决理发师问题的程序;

**理发师问题简述:**

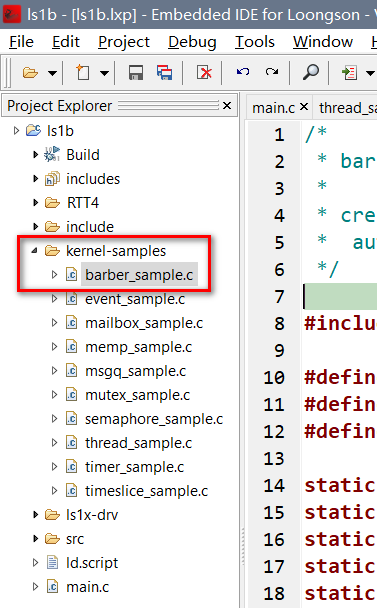
* 一个理发店由一个有几张椅子的等待室和一个放有一张理发椅的理发室组成。
* 若没有要理发的顾客，则理发师去睡觉；
* 若一顾客进入理发店，理发师正在为别人理发，且等待室有空椅子，则该顾客就找张椅子按顺序坐下；
* 若一顾客进入理发店，理发师在睡觉，则叫醒理发师为该顾客理发；
* 若一顾客进入理发店且所有椅子都被占用了，则该顾客就离开。

**伪码实现:**



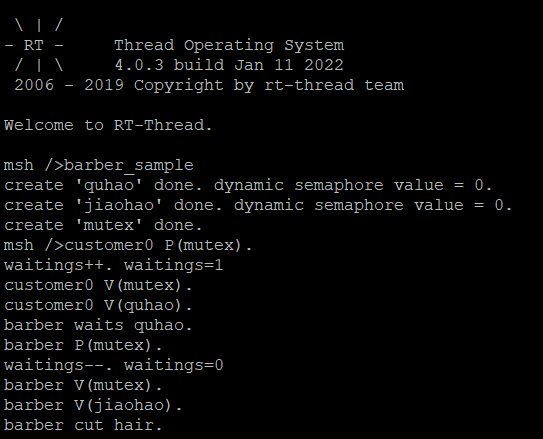
**用RT-thread的语法写出代码:**

在项目中的kernel-samples文件夹中新建barber\_sample.c文件;



/\*   
 \* barber\_sample.c   
 \*   
 \* created: 2022/1/12   
 \* author:   
 \*/   
   
#include <rtthread.h>   
   
#define THREAD\_PRIORITY 25   
#define THREAD\_STACK\_SIZE 1024   
#define THREAD\_TIMESLICE 5   
   
static rt\_sem\_t quhao = RT\_NULL; // 指向信号量的指针   
static rt\_sem\_t jiaohao = RT\_NULL; // 指向信号量的指针   
static rt\_mutex\_t mutex = RT\_NULL; // 指向互斥量的指针   
static rt\_uint8\_t waitings = 0; // 等待人数;   
static rt\_uint8\_t chairs = 10; // 空余椅子数;   
   
ALIGN(RT\_ALIGN\_SIZE)   
static char thread1\_stack[1024];   
static struct rt\_thread thread1;   
static void rt\_thread1\_entry(void \*parameter)   
{// thread1为理发师线程;   
 static rt\_err\_t result;   
 while(1)   
 {   
 result = rt\_sem\_take(quhao, RT\_WAITING\_FOREVER); // P()操作;   
 rt\_kprintf("barber waits quhao.\n");   
 if (result != RT\_EOK)   
 {   
 rt\_kprintf("barber P(quhao) failed.\n");   
 rt\_sem\_delete(quhao);   
 return;   
 }   
 else   
 {// 如果P到了quhao; 每条语句后的rt\_kprintf()语句起到了类似注释的作用;   
 rt\_mutex\_take(mutex, RT\_WAITING\_FOREVER); rt\_kprintf("barber P(mutex).\n");   
 waitings--; rt\_kprintf("waitings--. waitings=%d\n", waitings);   
 rt\_mutex\_release(mutex); rt\_kprintf("barber V(mutex).\n");   
 rt\_sem\_release(jiaohao); rt\_kprintf("barber V(jiaohao).\n");   
 rt\_kprintf("barber cut hair.\n");   
 rt\_thread\_mdelay(666);   
 }   
 }   
}   
   
ALIGN(RT\_ALIGN\_SIZE)   
static rt\_uint8\_t i;   
   
static void rt\_thread2\_entry(void \*parameter)   
{// thread2为顾客进程; 每条语句后的rt\_kprintf()语句起到了类似注释的作用;   
 static rt\_err\_t result;   
   
 rt\_mutex\_take(mutex, RT\_WAITING\_FOREVER); rt\_kprintf("customer%d P(mutex).\n", i);   
   
 if(waitings<chairs) {   
 waitings++; rt\_kprintf("waitings++. waitings=%d\n", waitings);   
 rt\_mutex\_release(mutex); rt\_kprintf("customer%d V(mutex).\n", i);   
 rt\_sem\_release(quhao); rt\_kprintf("customer%d V(quhao).\n", i);   
 result = rt\_sem\_take(quhao, RT\_WAITING\_FOREVER); // P(jiaohao)   
 rt\_kprintf("customer%d P(jiaohao).\n", i);   
 if (result != RT\_EOK) {   
 rt\_kprintf("customer%d P(jiaohao) failed.\n", i);   
 rt\_sem\_delete(jiaohao);   
 return;   
 }   
 else {// 如果P到了jiaohao   
 rt\_kprintf("customer%d has a hair cut.\n", i);   
 }   
 }   
 else {   
 rt\_mutex\_release(mutex); rt\_kprintf("customer%d V(mutex).\n", i);   
 }   
}   
   
/\* 信号量示例的初始化 \*/   
int barber\_sample()   
{   
 /\* 创建动态信号量quhao，初始值是0 \*/   
 quhao = rt\_sem\_create("quhao", 0, RT\_IPC\_FLAG\_FIFO);   
 if (quhao == RT\_NULL)   
 {   
 rt\_kprintf("create dynamic semaphore 'quhao' failed.\n");   
 return -1;   
 }   
 else   
 {   
 rt\_kprintf("create 'quhao' done. dynamic semaphore value = 0.\n");   
 }   
   
 /\* 创建动态信号量jiaohao，初始值是0 \*/   
 jiaohao = rt\_sem\_create("jiaohao", 0, RT\_IPC\_FLAG\_FIFO);   
 if (jiaohao == RT\_NULL)   
 {   
 rt\_kprintf("create dynamic semaphore 'jiaohao' failed.\n");   
 return -1;   
 }   
 else   
 {   
 rt\_kprintf("create 'jiaohao' done. dynamic semaphore value = 0.\n");   
 }   
   
 /\* 创建一个动态互斥量 \*/   
 mutex = rt\_mutex\_create("mutex", RT\_IPC\_FLAG\_FIFO);   
 if (mutex == RT\_NULL)   
 {   
 rt\_kprintf("create dynamic mutex failed.\n");   
 return -1;   
 }   
 else   
 {   
 rt\_kprintf("create 'mutex' done.\n");   
 }   
   
 rt\_thread\_init(&thread1,   
 "thread1",   
 rt\_thread1\_entry,   
 RT\_NULL,   
 &thread1\_stack[0],   
 sizeof(thread1\_stack),   
 THREAD\_PRIORITY, THREAD\_TIMESLICE);   
 rt\_thread\_startup(&thread1);   
   
 // 创建30个顾客线程;   
 for(i=0; i<30; i++)   
 {   
 // 设计两种顾客到来时间;   
 if(i%2==0)   
 rt\_thread\_mdelay(300);   
 else   
 rt\_thread\_mdelay(700);   
 struct rt\_thread thread2;   
 rt\_thread\_t tid;   
 /\* 创建线程\*/   
 tid = rt\_thread\_create("thread2",   
 rt\_thread2\_entry, RT\_NULL,   
 THREAD\_STACK\_SIZE,   
 THREAD\_PRIORITY-1, THREAD\_TIMESLICE);   
 /\* 如果获得线程控制块，启动这个线程 \*/   
 if (tid != RT\_NULL)   
 rt\_thread\_startup(tid);   
   
 return 0;   
 }   
}   
/\* 导出到 msh 命令列表中 \*/   
MSH\_CMD\_EXPORT(barber\_sample, barber sample);

**运行结果:**



**存在问题:**

不知道为什么, 理发师为第一个顾客理完发后, 程序会卡死在线程1的result = rt\_sem\_take(quhao, RT\_WAITING\_FOREVER);, 导致程序无法继续正常执行.