# 六、RT-Thread操作系统开发

## 1. 设计思路

### 1.1 功能描述

本次实验利用RT-Thread操作系统实现电影院的同步与互斥问题。  
假设电影院中有三部不同电影，每部电影一直循环播放，一直到没有观看该电影的观众为止才可以播放下一部电影。观众按照想要观看电影的类别分为三类，当一个观看某类电影观众进入电影院时，如果他是第一个该类电影的观众，则他可以要求放自己看的电影。

### 1.2 实现方法

每个观众可以用一个线程实现，每个观众线程有一个随机初始化的数值num（0<num<3)，用来标识所要看的电影。  
线程调度采用时间片方法，时间片长度固定。以此来实现线程的并发。  
运用信号量机制来实现线程之间的同步和互斥。包括一个同步信号量（用来实现放映电影的顺序）和三个互斥信号量（用来实现观看相同电影的观众对临界资源count[id]的互斥访问，count[id]记录了前来观看序号为id的电影的观众人数）。

* c语言伪代码

threadi//观看i号电影的观众线程  
{  
 P(mi);  
 counti++;  
 if(counti==1)  
 {  
 P(movie)  
 }  
 V(mi);  
 WatchMovie()；  
 P(mi)  
 counti--;  
 if(counti==0)  
 {  
 V(movie);  
 }  
 V(mi);  
 Leave();  
}

## 2. 实现代码

* 信号量指针和临界资源的定义

#define THREAD\_PRIORITY 25  
#define THREAD\_TIMESLICE 5  
#define THREAD\_STACK\_SIZE 1024  
/\* 指向信号量的指针 \*/  
static rt\_sem\_t movie=RT\_NULL;//表示电影的信号量，为同步信号量  
ALIGN(RT\_ALIGN\_SIZE)  
static rt\_mutex\_t mutex[3];//互斥信号量  
static rt\_uint32\_t count[3];//每种电影正在观看的人数，需要初始化为0

* 观众线程实现

static void rt\_thread\_entry(void \*parameter)/\*观众线程\*/  
{  
 rt\_err\_t result;  
 rt\_uint32\_t id=(rt\_uint32\_t)parameter;  
 /\*获取参数，id为该观众观看电影的序号\*/  
 rt\_mutex\_take(mutex[id], RT\_WAITING\_FOREVER);  
 /\*因为count[id]是互斥资源，所有需要用互斥信号量\*/  
 count[id]++;  
 rt\_kprintf("A customer comes in to watch movie%d.\n",id);  
 if(count[id]==1)  
 {  
 result = rt\_sem\_take(movie, RT\_WAITING\_FOREVER);  
 /\* 永久方式等待信号量,即必须等待上一个观看上一个电影的观众走完才能放新电影 \*/  
 if (result != RT\_EOK)  
 {  
 rt\_kprintf(" failed.\n");  
 rt\_sem\_delete(movie);  
 return;  
 }  
 rt\_kprintf("Movie%d. starts now\n",id);  
 }  
 rt\_mutex\_release(mutex[id]);  
 //看电影  
 int delayTime;  
 delayTime = rand()%200+20;  
 rt\_thread\_mdelay(delayTime);  
 rt\_mutex\_take(mutex[id], RT\_WAITING\_FOREVER);  
 /\*因为count[id]是互斥资源，所有需要用互斥信号量\*/  
 count[id]--;  
 rt\_kprintf("A customer finishes watching movie%d and leave.\n",id);  
 if(count[id]==0)  
 {  
 rt\_kprintf("Movie%d. ends now\n",id);  
 rt\_sem\_release(movie);  
 }  
 rt\_mutex\_release(mutex[id]);  
}

* movie\_theater\_sample函数（创建线程和初始化信号量）

int movie\_theater\_sample()  
{  
 /\* 信号量的初始化 \*/  
 movie=rt\_sem\_create("moviesem", 1, RT\_IPC\_FLAG\_FIFO);//movie初始化为1  
 if (movie == RT\_NULL)  
 {  
 rt\_kprintf("create dynamic semaphore failed.\n");  
 return -1;  
 }  
 int i=0;  
 for(i=0;i<3;i++)  
 {  
 mutex[i]=RT\_NULL;  
 mutex[i] = rt\_mutex\_create("dmutex", RT\_IPC\_FLAG\_FIFO);  
 if (mutex[i] == RT\_NULL)  
 {  
 rt\_kprintf("create dynamic semaphore failed.\n");  
 return -1;  
 }  
 }  
 /\*计数器的初始化\*/  
 for(i=0;i<3;i++)  
 count[i]=0;  
 for(i=0;i<10;i++)/\*创建10个观看者线程\*/  
 {  
 rt\_thread\_mdelay(10);  
 struct rt\_thread thread;  
 rt\_thread\_t tid;  
 int num;  
 num = rand();  
 /\* 创建线程,假设观看movie1,2,3的人随机前来\*/  
 tid = rt\_thread\_create("thread",  
 rt\_thread\_entry, (void\*)(num%3),  
 THREAD\_STACK\_SIZE,  
 THREAD\_PRIORITY, 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;  
}

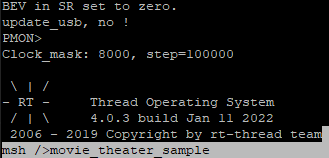
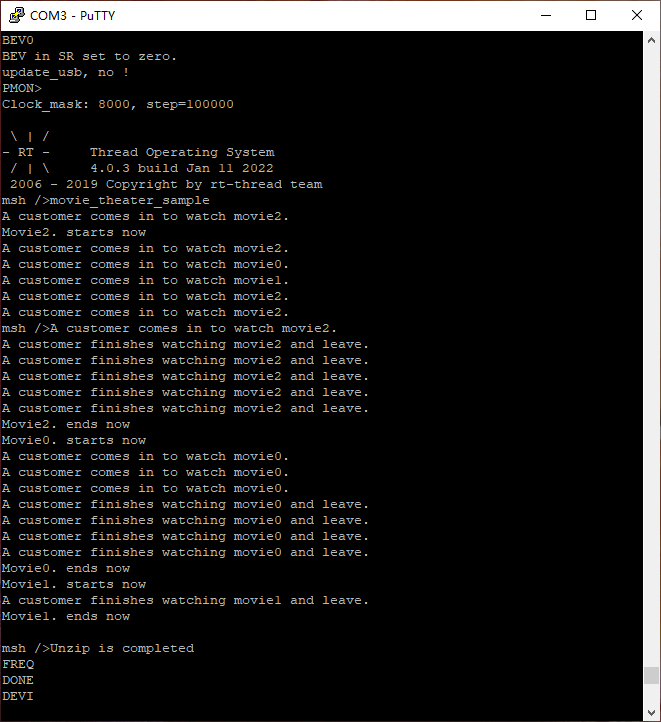
* 加入到msh命令中

MSH\_CMD\_EXPORT(movie\_theater\_sample, movie theater);

## 3. 调试过程

打卡PuTTY,输入命令

movie\_theater\_sample

如下图：  
  
输出结果展示  


## 4. 遇到问题

* 因为采用时间片调度，而输出操作并不是原子操作，所以有的时候输出到一半会被打断  
  解决方式：随机调整线程的delay时间进行避退，尽量减少输出被打断的可能。