直播间问题 实验报告

陈家豪 19307130210

回答问题

Question 1:

用户对直播间资源的占用可抽象成变量user_num[3]，表示观看3部影片的用户人数，各用户线程和直播间线程会共同访问这个变量，新用户排队使其增加，离开使其减少，直播间需要访问该变量以确定播放还是停止。线程间需互斥访问user_num[3]。

可以有无数用户线程同时观看一部影片。

Question 2:

3个线程：

直播间线程：

在设定最大线程运行时间内{

等待第一个用户进场

循环1 ~ 3部影片{

查看排队看影片i的人数：

若user_num[i]!=0{

将影片i设为播放状态，允许用户进入

电影放映时间内{

每隔1s检查一次在看人数，若为0则break

}

影片时间到或没人看，停止播放

等待所有观看者退出

}

若user_num[i]==0，不做任何事

检查是否有人在排队，若有人则继续播放下一部，没人就退出循环

}

}

用户线程：

等待直播间线程开放排队入口

在相应影片排队，将user_num[i]+1

等待直播间线程开放进入直播间的入口

观看影片，随机设置观看时间

若观看时间到或影片结束放映，用户退出,user_num[i]-1

user_manager线程：

在线程最大运行时间内{

随机生成观看影片标签

创建用户线程

随机等待数秒

}

代码解释

1. 用到的sem_t信号量：

```
sem_t mutex1;
sem_t mutex2;
sem_t mutex_print;
sem_t mutex_user[3];
sem_t mutex_film[3];
sem_t mutex_out[3];
```

mutex1:

初值为0，0表示没有用户在排队，非0表示有用户在排队，用于启动直播间线程播放影片；

user线程在成功排队后，`sem_post(&mutex1)`，在退出后，`sem_wait(&mutex1)`

broadcast线程会等待user线程启动

```
void broadcast(){
    while(time(NULL)<end_time){ //达到设定最大时间结束放映
        sem_wait(&mutex1);//等待第一个用户进场，开始循环放映
        sem_post(&mutex1);
        ……
```

mutex2:

初值为1，用于阻塞user线程排队，设为0时，user不能排队，即不能访问user_num[]，设为1时可以

```
void user(void * arg){

    sem_wait(&mutex2);
    sem_post(&mutex2);
    ……
```

在broadcast线程结束播放影片，等待所有用户离开时，需禁止新用户排队，否则有死锁风险；在现有用户都离开后，可以再次开放排队入口

```
//将该影片设为停止状态
sem_wait(&mutex2);  //禁止新用户排队
film_state[i]=0;    //将file_state[i]设为0后，现有观众会自行退出
sem_wait(&mutex_print);
getDateTime();
if(time(NULL)>=cur_end_time)
printf("影片播放完毕，");
printf("影片%d结束放映\n",i+1);
sem_post(&mutex_print);

sem_wait(&mutex_out[i]);    //等待正在观看i的观众全部离开
sem_post(&mutex_out[i]);
sem_wait(&mutex_film[i]);   //关闭用户进入直播间的入口
sem_post(&mutex2);
```

mutex_print:

在打印信息时，有时会用到多个printf()函数，防止中间被插入其他线程的printf()，初值为1

mutex_user[3]:

对user_num[3]的互斥锁，初值为1

mutex_film[3]:

是用户进入直播间的关卡，初值为0，broadcast线程在播放影片i时，将mutex_film[i]设为1，user线程可以进行下一步，否则被阻塞在sem_wait(&mutex_film[i]).

```
// 用户等待该纪录片播放
    sem_wait(&mutex_film[film]);    //等待broadcast线程将mutex_file[i]变为1
    sem_wait(&mutex_print);
```

mutex_out[3]:

用于实现broadcast线程在结束播放后等待所有用户离开；初值为1

mutex_out[i]=1时，表示没人在看影片i；为0时表示有人在看影片i

user线程，第一个排队看i的人将mutex_out[i]设为0；最后一个离开的人将其设为1；

broadcast线程在结束放映后会`sem_wait(&mutex_out[i])`，直到最后一个user离开

2.主线程

初始化信号量；

设置线程运行时间；

启动broadcast线程，user_manager线程；

回收线程；

3.broadcast线程

截取了主要代码：注释说明了运行细节

```
void broadcast(){
    while(time(NULL)<end_time){ //达到设定最大时间结束放映

        sem_wait(&mutex1);//等待第一个用户进场，开始循环放映
        sem_post(&mutex1);

        for(i=0;1;i = (i+1)%3){// i 表示当前放映影片序号

            //检测影片 i 是否有人要看
            sem_wait(&mutex_user[i]);
            if(user_num[i]!=0){//有人要看影片i，则放映
                sem_post(&mutex_user[i]);   //查询完，不需要占用锁了

                //将该影片设为播放状态
                film_state[i]=1;
                cur_film = i;
                sem_post(&mutex_film[i]);   //开放直播间入口，相应用户可以进入直播间

                int cur_end_time = time(NULL)+film_time[i]; //计算影片结束时间
                while(time(NULL)<cur_end_time){ //放映过程，时间到了结束放映
                    //每秒检测一次观影人数
                    sem_wait(&mutex_user[i]);
                    int cur_user_num = user_num[i];
                    if(cur_user_num==0){//无人观看，退出
                        sem_post(&mutex_user[i]);
                        break;
                    }
                    else{
                        sem_post(&mutex_user[i]);
                    }
                    sleep(1);
                }

                //将该影片设为停止状态
                sem_wait(&mutex2);  //禁止新用户排队
                film_state[i]=0;            //将file_state[i]设为0后，现有观众会自行退出

                sem_wait(&mutex_out[i]);    //等待正在观看i的观众全部离开
```

```
                sem_post(&mutex_out[i]);    //若没有禁止新用户排队，可能一直有人排队进来又被赶出去
                                            //新用户让&mutex_out[i]变为0，将broadcast阻塞在这里

                sem_wait(&mutex_film[i]);   //关闭用户进入直播间的入口
                sem_post(&mutex2);

            }

            else{       //没人看，则不放映
                sem_post(&mutex_user[i]);
            }

            //检测是否还有人在排队，没人看直播间就break，直播间休息，再次等待新人到来
            int flag = if_have_user();
            if(flag==0)
                break;
        }

    }
}
```

4.user_manager线程

就是user线程的发射器，创建的user线程不用回收

```
void user_manager(){
    pthread_t p;
    pthread_attr_t attr;
    pthread_attr_setdetachstate(&attr, 1);

    while(time(NULL) < end_time){
        // 随机生成纪录片序号
        int * index = (int *) malloc (sizeof(int));
        *index = rand()%3;
        pthread_create(&p, &attr, (void *)user, (void *)index);
        sleep(rand() % USER_MAX_ENTER);
    }
    sleep(1);
}
```

5.user线程

```
void user(void * arg){

    sem_wait(&mutex2);      //在broadcast清除用户的时候，mutex2阻塞，不能排队
    sem_post(&mutex2);

    //用户加入排队队列
    int film = *((int *)arg);
    sem_wait(&mutex_user[film]);    //访问user_num[i]
```

```
    if(user_num[film]==0){       //mutex_out[i]的作用为broadcast进程等待所有观众离开后再切换影片
        sem_wait(&mutex_out[film]); //mutex_out[i]为0时表示还有观众，为1时表示观众全部退出
    }                               //第一个观众进入让其变为0，最后一个退出时将其变为1
    user_num[film] ++;              //排队人数加1
    sem_post(&mutex_user[film]);

    sem_post(&mutex1);              //每进一个人就让mutex+1，退出时-1,表示还有没有人在看，控制直播间休息还是继续运行

    // 用户等待该纪录片播放，进入直播间
    sem_wait(&mutex_film[film]);    //等待broadcast进程将mutex_file[i]变为1
    sem_post(&mutex_film[film]);

    //观看过程
    int user_time = time(NULL)+rand()%(USER_MAX_WATCH-USER_MIN_WATCH)+USER_MIN_WATCH;
    while(time(NULL)<user_time&&film_state[film]);  //用户观看时间到或影片结束，用户退出

    //用户退出
    sem_wait(&mutex_user[film]);
    user_num[film]--;       //人数-1
    if(user_num[film]==0){
        sem_post(&mutex_out[film]); //最后一个退出的将mutex_out[i]变1，作用如上所述
    }
    sem_post(&mutex_user[film]);
    sem_wait(&mutex1);  //mutex-1
}
```

运行结果

设3部电影时长都为10s，线程运行时长30s，0～3s的间隔有一个新用户，用户观看时长1～10s

以下截取部分运行结果进行说明

在broadcast线程循环放映影片时，每隔1s打印一次在线人数和放映电影

开始阶段：

```
15:19:51        broadcast thread starting
15:19:51        user thread starting
15:19:51        直播间休息中，等待用户进入
15:19:51        影片2排队+1
15:19:51        影片2开始放映
15:19:51        当前影片：2     在线人数：    影片1: 0     影片2: 1     影片3: 0
15:19:51        1个用户进入直播间2
15:19:52        1个用户退出直播间2，直播间剩余人数：0
15:19:52        当前影片：2     在线人数：    影片1: 0     影片2: 0     影片3: 0
15:19:52        影片2已无人观看
15:19:52        影片2结束放映
15:19:52        直播间休息中，等待用户进入
15:19:52        影片2排队+1
```

```
15:19:52        影片2开始放映
15:19:52        当前影片：2     在线人数：    影片1: 0     影片2: 1     影片3: 0
15:19:52        1个用户进入直播间2
15:19:53        当前影片：2     在线人数：    影片1: 0     影片2: 1     影片3: 0
15:19:54        当前影片：2     在线人数：    影片1: 0     影片2: 1     影片3: 0
15:19:54        影片2排队+1
```

4行起，第一个用户排队，直播间开始放映相应影片，之后该用户可以进入直播间

8行，用户退出，没人在线导致直播间关闭。

13行，又有人排队，直播间再次启动。

影片切换过程：

```
15:20: 1        当前影片：2     在线人数：    影片1: 2     影片2: 1     影片3: 3
15:20: 1        影片3排队+1
15:20: 2        影片播放完毕，影片2结束放映
15:20: 2        1个用户退出直播间2，直播间剩余人数：0
15:20: 2        影片3开始放映
15:20: 2        当前影片：3     在线人数：    影片1: 2     影片2: 0     影片3: 4
15:20: 2        1个用户进入直播间3
15:20: 2        1个用户进入直播间3
15:20: 2        1个用户进入直播间3
15:20: 2        1个用户进入直播间3
15:20: 3        当前影片：3     在线人数：    影片1: 2     影片2: 0     影片3: 4
15:20: 3        影片1排队+1
15:20: 3        影片1排队+1
15:20: 4        1个用户退出直播间3，直播间剩余人数：3
15:20: 4        1个用户退出直播间3，直播间剩余人数：2
```

3行，播放时间到，影片结束，还有一个人再看，直播间等待其离开后，开始放映影片3.

6行，之前已有4人在排队看影片3，现在显示4人进入直播间

结束阶段：

```
15:20:22        user thread ending, no more new user
15:20:22        影片播放完毕，影片1结束放映
15:20:22        1个用户退出直播间1，直播间剩余人数：2
15:20:22        1个用户退出直播间1，直播间剩余人数：1
15:20:22        1个用户退出直播间1，直播间剩余人数：0
15:20:22        影片2开始放映
15:20:22        当前影片：2     在线人数：    影片1: 0     影片2: 4     影片3: 4
15:20:22        1个用户进入直播间2
15:20:22        1个用户进入直播间2
15:20:22        1个用户进入直播间2
15:20:22        1个用户进入直播间2
15:20:23        当前影片：2     在线人数：    影片1: 0     影片2: 4     影片3: 4
```

线程运行时间到，user_manager线程不再发出新的user线程。

出于人性化考虑，直播间会让目前还在等待的人看完电影。

```
15:20:38        当前影片：3     在线人数：    影片1: 0    影片2: 0    影片3: 1
15:20:39        当前影片：3     在线人数：    影片1: 0    影片2: 0    影片3: 1
15:20:40        1个用户退出直播间3，直播间剩余人数：0
15:20:40        当前影片：3     在线人数：    影片1: 0    影片2: 0    影片3: 0
15:20:40        影片3已无人观看
15:20:40        影片3结束放映
15:20:40        broadcast thread ending
```

结束