操作系统原理及Linux内核

西安邮电大学

读者——写者问题&哲学家进餐问题

读者--写者问题

问题描述

一个数据集（如文件）被多个并发进程(线程)共享，一些进程(线程)只读取数据集内容（读者），而另一些进程(线程)则只修改数据集内容（写者）。访问规则：

- 允许多个读者同时读取数据集；当有读者读数据集时，不允许任何写者进程写。
- 当一个写者进程写数据集期间，不允许任何其它写者进程写，也不允许任何读者读。

即“读－写”互斥，“写－写”互斥，“读－读”允许

读者--写者问题

问题分析

即 “读一写”互斥，“写一写”互斥，“读一读”允许

变量：readcount, 读者数

信号量： **sdata, 数据集，初值1；** **srcount, 读者数变量，初值1；**

读者--写者问题

```
void * reader(void *p){
    sem_wait(&srcount);
    readcount = readcount +1;
    if (1==readcount)
        sem_wait(&sdata);
    sem_post(&srcount);
    Reading dataset ….
    sem_wait(&srcount);
    readcount = readcount - 1;
    if (0==readcount)
        sem_post(&sdata);
    sem_post(&srcount);
    return NULL;
}
```

```
void * writer(void *p){
    sem_wait(&sdata);
    Writing dataset ….
    sem_post(&sdata);
    return NULL;
}
```

存在问题：

写者进程在实际应用中一般拥有更高操作优先权，但有可能长期推迟。

读者--写者问题(写者优先)


```
wait(sdata);
For i=1 to N
    wait(sread);
写数据集
For i=1 to N
    signal(sread);
signal(sdata);
```

sdata, 初值1；

读者数量

sread, 初值N；

```
wait(sdata);
wait(sread);
signal(sdata);
读数据集
signal(sread);
signal(sdata);
```

读者—写者问题(写者优先)

```
#define M 10
void * writer(void *p){
    sem_wait(&sdata);
    for(int i=0; i<N; i++ )
        sem_wait(&sread);
    Writing dataset
    return NULL;
}
void * reader(void *p){
    sem_wait(&sdata);
    sem_wait(&sread);
    sem_post(&sdata);
    Reading dataset
}
NULL, reader, NULL);
.....
}
```

Swait(sread, N, 0)

Swait(sdata, 1, 0, sread, 1, 1)

读者—写者问题(信号量集)

```
#define M 10
#define N 20
sem_t sdata, sread;
void * writer(void *p){
    Swait(sdata, 1, 1)
    Swait(sread, N, 0);
    Writing dataset ….
    Ssignal (sdata, 1);
    return NULL;
}
        pthread_create(&tid2[i], NULL, reader, NULL);
    sem_destroy(&sdata);
    ……
}
```

Swait(sdata, 1,1, sread, N, 0)
???

```
    Reading dataset ….
    Ssignal(sread, 1);
    return NULL;
}
```

哲学家进餐问题

问题描述（1965年Dijkstra）

5个哲学家围坐在一张圆桌周围，桌子上有意大利通心粉，相邻两个哲学家之间只有一把叉子。通心粉很滑，需要两把叉子才能吃。

哲学家的生活包含思考和吃饭两项活动。当一个哲学家觉得饥饿时，需要分别拿起他左、右两侧的叉子，每次限拿一把，不分次序。如成功拿到两把叉子，就开始吃饭，吃完以后放下叉子继续思考。

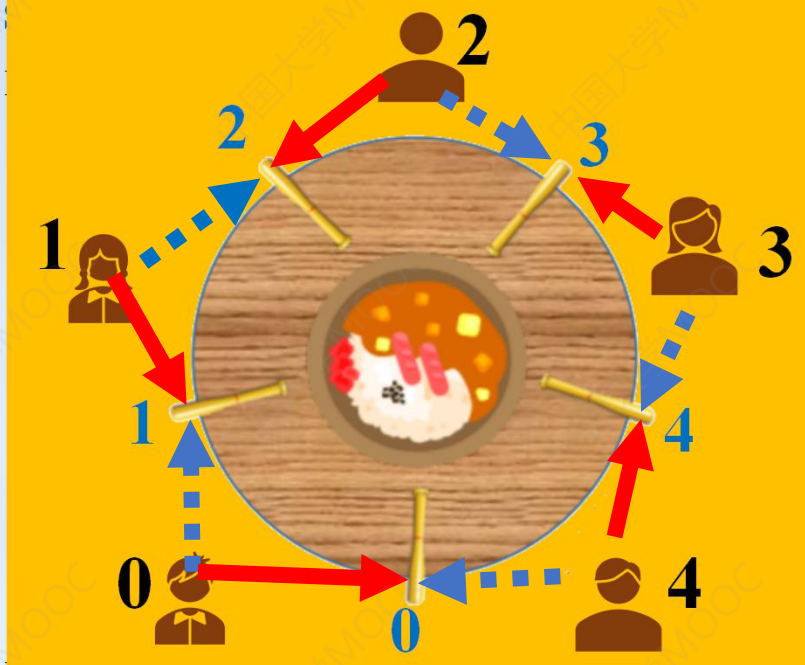
哲学家进餐问题

问题描述（1965年Dijkstra）

信号量：

sfork[5], 0—4叉子

哲学家进餐问题



死锁(Deadlock)

```
void *philosopher (void *p){
   int id =(int)p;
   while(1){
        Thinking ……..
        sem_wait(&sfork[id]);
        sem_wait(&sfork[(id+1)%5]);
        Eating ……
        sem_post(&sfork[id]);
        sem_post(&sfork[(id+1)%5]);
    }
    return NULL;
}
```

哲学家进餐问题

```
void *philosopher (void *p){//方案1
  int id =(int)p;
  while(1){
      Thinking ……..
      if(id%2==0){
          sem_wait(&sfork[id]);
          sem_wait(&sfork[(id+1)%5]);
      }else{
          sem_wait(&sfork[(id+1)%5]);
          sem_wait(&sfork[id]);
      }
      Eating ……
      sem_post(&sfork[id]);
      sem_post(&sfork[(id+1)%5]);
   } return NULL;
}
```

```
void *philosopher (void *p){//方案2
  int id =(int)p;
  while(1){
      Thinking ……..
      if(id==4){
          sem_wait(&sfork[0]);
          sem_wait(&sfork[4);
      }else{
          sem_wait(&sfork[id]);
          sem_wait(&sfork[id+1]);
      }
      Eating ……
      sem_post(&sfork[id]);
      sem_post(&sfork[(id+1)%5]);
   } return NULL;
}
```

哲学家进餐问题（ADD型信号量）

```
void *philosopher (void *p){
   int id =(int)p;
   while(1){
        Thinking ……...
        Swait(sfork[id], sfork[(id+1)%5]);
        Eating ……
        Ssignal (sfork[id], sfork[(id+1)%5]);
    }
    return NULL;
}
```

```
l*) i);
```

小结

读者-写者&哲学家进餐问题

读者-写者

- 读-写，写-写互斥；读-读允许
- 方案1：第1个读者抢占数据集；最后1个离开释放数据集
- 方案2：设定读者上限

小结

读者-写者&哲学家进餐问题

哲学家进餐

- 破坏循环等待
- 方案1：偶数先右后左，基数相反；
- 方案2：4号哲学家先左后右，其他相反

解决策略

- 寻找临界资源
- 定义信号量
- 使用前申请
- 使用后释放

字体：中文：思源黑体≥24
英文：新罗马 ≥24

配色

层级  文件管理

特殊字体双击安装