1.3

Contents

- P/V

P/V的作用

- 保护共享对象
- 逻辑控制

保护共享对象

- P(mutex); Modify(sharedObject); V(mutex);
- 简单的例子：简单的独木桥问题
- 一个南北向的独木桥，一次只能有一个人通过，现在要控制两边的人按顺序过桥。

- 只要给桥加一个保护共享对象的信号量即可

保护共享对象

- P(mutex); Modify(sharedObject); V(mutex);
- 简单的例子：暴力实现生产者消费者问题
  - 生产者消费者问题本质是每个人都想改共享的buffer
- 只用一个信号量，保护整个buffer即可

- 有什么问题？
  - Buffer有可能满，也有可能没东西
  - Buffer满了会怎么样？没东西会怎么样？

逻辑控制

- P可以表示：若条件不满足，就排队等待。
- V可以表示：现在条件满足了！通知排队的第一个人放行。
- 现在实现生产者消费者问题：加信号量分别表示“buffer没空了，生产者要等着！”，“buffer没东西，消费者要等着！”
  - 要注意P的顺序！

逻辑控制

```
/* Insert item onto the rear of shared buffer sp */
void sbuf_insert(sbuf_t *sp, int item)
{
    P(&sp->slots);                  /* Wait for available slot */
    P(&sp->mutex);                  /* Lock the buffer         */
    if (++sp->rear >= sp->n)        /* Increment index (mod n) */
        sp->rear = 0;
    sp->buf[sp->rear] = item;       /* Insert the item         */
    V(&sp->mutex);                  /* Unlock the buffer       */
    V(&sp->items);                  /* Announce available item */
}
```

sbuf.c

逻辑控制

——交换了橙色方框里的语句会怎么样？

```
/* Insert item onto the rear of shared buffer sp */
void sbuf_insert(sbuf_t *sp, int item)
{
    P(&sp->slots);                  /* Wait for available slot */
    P(&sp->mutex);                  /* Lock the buffer         */
    if (++sp->rear >= sp->n)        /* Increment index (mod n) */
        sp->rear = 0;
    sp->buf[sp->rear] = item;       /* Insert the item         */
    V(&sp->mutex);                  /* Unlock the buffer       */
    V(&sp->items);                  /* Announce available item */
}
```

sbuf.c

P/V应用：再看第一类读写者问题

mutex: 保护共享对象 *readcnt*
w: 保护共享对象 + 逻辑控制

Readers:

```
int readcnt;     /* Initially 0 */
sem_t mutex, w; /* Both initially 1 */

void reader(void)
{
  while (1) {
    P(&mutex);
    readcnt++;
    if (readcnt == 1) /* First in */
      P(&w);
    V(&mutex);

    /* Reading happens here */

    P(&mutex);
    readcnt--;
    if (readcnt == 0) /* Last out */
      V(&w);
    V(&mutex);
  }
}
```

Writers:

```
void writer(void)
{
  while (1) {
    P(&w);

    /* Writing here */

    V(&w);
  }
}
```

rw1.c

Arrivals: R1 R2 W1 R3

P/V应用：第一类读写者问题的变式

- 一般的独木桥问题
- 一个东西向的独木桥，可以通过很多很多人（不做限制）
- 但是两边不能同时有人上桥，该怎么办？

一般的独木桥问题

```
void west(){
        P(&mutex_w);
        num_west ++;
        if (num_west == 1)
                P(&bridge);
        V(&mutex_w);

        cross_bridge();

        P(&mutex_w);
        num_west --;
        if (num_west == 0)
                V(&bridge);
        V(&mutex_w);
}
```

```
void east(){
        P(&mutex_e);
        num_east ++;
        if (num_east == 1)
                P(&bridge);
        V(&mutex_e);

        cross_bridge();

        P(&mutex_e);
        num_east --;
        if (num_east == 0)
                V(&bridge);
        V(&mutex_e);
}
```

P/V应用：第一类读写者问题的变式

- 一般的独木桥问题 的变式
- 一个东西向的独木桥，桥上最多只能同时有5个人
- 而且两边不能同时有人上桥，该怎么办？

一般的独木桥问题的变式

```
void west(){
        P(&mutex_w);
        num_west ++;
        if (num_west == 1)
                P(&bridge);
        V(&mutex_w);
        P(&capacity);
        cross_bridge();
        V(&capacity);
        P(&mutex_w);
        num_west --;
        if (num_west == 0)
                V(&bridge);
        V(&mutex_w);
}
```

```
void east(){
        P(&mutex_e);
        num_east ++;
        if (num_east == 1)
                P(&bridge);
        V(&mutex_e);
        P(&capacity);
        cross_bridge();
        V(&capacity);
        P(&mutex_e);
        num_east --;
        if (num_east == 0)
                V(&bridge);
        V(&mutex_e);
}
```

P/V应用：第一类读写者问题的问题

- 读者源源不断的来，写者永远写不了，一直等着
  - 写者会饥饿
- 第二类读写者问题——写者优先
  - 只要有写者来了，就P一个信号量，告诉新的读者要等待
  - 所有写着都走了，才V这个信号量，让新的读者进来。

相比于第一类读写者问题，这里多了两个新的信号量：
wmutex: 用来保护writecnt这个变量
r: 用来控制“让读者等待”这个逻辑

```
int readcnt, writecnt;       // Initially 0
sem_t rmutex, wmutex, r, w; // Initially 1
void reader(void)
{
  while (1) {
    P(&r);
    P(&rmutex);
    readcnt++;
    if (readcnt == 1) /* First in */
      P(&w);
    V(&rmutex);
    V(&r)

    /* Reading happens here */

    P(&rmutex);
    readcnt--;
    if (readcnt == 0) /* Last out */
      V(&w);
    V(&rmutex);
  }
}
```

```
void writer(void)
{
  while (1) {
    P(&wmutex);
    writecnt++;
    if (writecnt == 1)
        P(&r);
    V(&wmutex);

    P(&w);
    /* Writing here */
    V(&w);

    P(&wmutex);
    writecnt--;
    if (writecnt == 0);
        V(&r);
    V(&wmutex);
  }
}
```

第二类读写者问题的变式

- 难一些的独木桥问题
- 一个东西向的独木桥，桥上人数不限
- 两边不能同时有人上桥
- 要避免饥饿的问题出现！

- 思路：类似读者写者问题，但是“写者”之间并不互斥

```
semaphore mutex_w = 1;
semaphore mutex_e = 1;
semaphore transfer = 1;
semaphore rope = 1;

int num_west = 0;
int num_east = 0;
```

```
void west(){
        P(&transfer);
        P(&mutex_w);
        num_west ++;
        if (num_west == 1)
                P(&rope);
        V(&mutex_w);
        V(&transfer);

        baboon_cross_gorge();

        P(&mutex_w);
        num_west --;
        if (num_west == 0)
                V(&rope);
        V(&mutex_w);
}
```

```
void east(){
        P(&transfer);
        P(&mutex_e);
        num_east ++;
        if (num_east == 1)
                P(&rope);
        V(&mutex_e);
        V(&transfer);

        baboon_cross_gorge();

        P(&mutex_e);
        num_east --;
        if (num_east == 0)
                V(&rope);
        V(&mutex_e);
}
```

好消息！

- 考试应该不会这么复杂
- 2013：交通管理系统，3个信号量，6行代码
- 2014：家人吃水果：3个信号量，6行代码