2020 年春季学期操作系统第一次作业：多缓冲区，水果，管程

2018 级计算机科学与技术（中美合作）

包亦航
2018011890

2020 年 4 月

问题一：

N 个共享缓冲区读写问题，我们现在有 N 个共享缓冲区，一个写进程，一个打印进程，需要我们设计代码来实现这个过程

在 Example_Program 目录下使用 python 对多共享缓冲区进行了实现，在该演示中，默认共享区大小是 5，默认有五个打印进程一个写进程

解答方案一（使用上课讲的新增临界资源的方法）

首先判断这是个同步互斥问题，有两个进程：一个写进程，一个打印进程

我们使用process变量来记录缓冲区中目前有的进程数，mutex用来保护该变量

```
int Process=0;
Semaphore mutex=1;

Write()
{
    while(1)
    {
        Prepare data

        P(mutex);
        if (process<N)
        {
            Process = Process+1;

            Write data into the memory

            V(mutex);
        }
       else
       {
            V(mutex); //体现了让权等待的原则
       }
    }
}

Print()
{
   while(1)
    {
        P(mutex);
        If (process>0)
        {
            Process = Process-1;

            Read data out

            V(mutex);

            Print data
        }
        else
        {
            V(mutex); //体现了让权等待的原则
```

```
        }
    }
}
int main()
{
    Process = 0
    mutex = 1
    cobegin
        Write()
        Print()
    coend
}
```

解答方案二（使用纯信号量方法）：

这是个同步互斥问题
我们使用use信号量记录缓冲区是否被占用，用available记录还有多少剩余空间，用record记录目前有几个被占用

```
Semaphore use=1; available=N; record=0;
write()
{
    while(1)
    {
        Prepare data

        P(available); P(use); //有空余再请求使用临界区

        Write data into the memory

        V(record);V(use); //先增加内容再释放临界区
    }
}

print()
{
    while(1)
    {
        P(record); P(use); //有内容再请求使用临界区

        Read data from memory

        V(available);V(use);  //先减少内容再释放临界区

       Print data
    }
}
int main()
{
    use=1;
    available=N;
    record=0;
    cobegin
        write()
        print()
    coend
}
```

问题二：桌上有个能盛下 N 个水果的空盘子。父亲不停地向盘中放苹果或桔子，儿子不停地从盘中取出桔子，女儿不停地从盘中取出苹果。规定 3 人不能同时从盘中取放水果。试用信号量实现 3 人代表的进程之间的同步。

解答方案：

这是一个同步互斥问题

我们这里的信号量设置方式如下：设置一个plate信号量，用来标记这个盘子是否被占用；设置一个cap信号量，用来标记盘子剩余的空间；设置一个apple信号量，用来标记盘子里有几个苹果；设置一个orange信号量，用来标记盘子里有几个橘子

代码如下：

```
1.  Semaphore plate=1, cap=N, apple=0, orange=0;
2.  Dad()
3.  {
4.      while(1)
5.      {
6.
7.          Prepare an apple;
8.
9.          P(cap); P(plate); //先看有没有剩余的空间再争夺盘子的使用权
10.
11.         Put the apple on the plate;
12.
13.         V(apple); V(plate); //先放苹果后释放盘子的使用权
14.
15.         Prepare an orange;
16.
17.         P(cap); P(plate); //先看有没有剩余的空间再争夺盘子的使用权
18.
19.         Put the orange on the plate;
20.
21.         V(orange); V(plate); //先放橘子后释放盘子的使用权
22.     }
23. }
24.
25. Son()
26. {
27.     While(1)
28.     {
29.         P(orange); P(plate) //先看有没有橘子再争夺盘子的使用权
30.
31.         Take an orange from the table;
32.
33.         V(cap);V(plate); //先拿走橘子后释放盘子的使用权
34.
35.         Eat the orange;
36.     }
37. }
38. Daughter()
39. {
40.     While(1)
41.     {
42.         P(apple); P(plate) //先看有没有苹果再争夺盘子的使用权
43.
```

```
        Take an apple from the table;

        V(cap);V(plate); //先拿走苹果后释放盘子的使用权

        Eat the apple;
    }
}
int main()
{
    plate=1;
    cap=N;
    apple=0;
    orange=0;
    cobegin
        Dad();
        Son();
        Daughter();
    coend
}
```

问题三：利用管程解决多读者多写者问题

解答方案1（读者优先）：

```
class Monitor_reader_writer()
{
    private:
        int reader_num;
        bool writer; //标记是否有写者在写
        condition read_q, write_q; //一个读队列一个写队列

        Monitor_reader_writer() //构造函数
        {
            reader_num = 0;
            writer = false
        }

    public:
        void start_read()
        {
            if (writer) read_q.wait(); //如果有写者在写就等待
            reader_num ++;
            if (read_q.queue) read_q.signal(); //如果有读者等待，将其唤醒
        }

        void end_read()
        {
            reader_num --;
            if (reader_num==0) write_q.signal(); //如果没有读者了，唤醒写者
        }

        void start_write()
        {
            If (reader_num>0 || writer) write_q.wait(); //如果有读者在读或有写者在写，等待
            writer = true; //如果可以了就标明正在写
        }
```

```
33.
34.         void end_write()
35.         {
36.             writer = false;
37.             if (read_q.queue) read_q.signal(); //如果有读者等优先读者
38.             else if (write_q.queue) write_q.signal(); //否则就唤醒写者
39.     } Mon
40. //----------以上是管程部分，接下来是读者和写者----------
41. void Read()
42. {
43.     while(1)
44.     {
45.         Mon.start_read();
46.
47.         reading from memory
48.
49.         Mon.end_read();
50.     }
51. }
52.
53. void Write()
54. {
55.     while(1)
56.     {
57.         Mon.start_write();
58.
59.         writing into memory
60.
61.         Mon.end_write();
62.     }
63. }
64.
65. void main()
66. {
67.     cobegin
68.      Read();
69.      Writ();
70.     coend
71. }
```

解答方案2（写者优先）：

```
1.  class Monitor_reader_writer()
2.  {
3.      private:
4.          int reader_num;
5.          bool writer; //标记是否有写者在写
6.          condition read_q, write_q; //一个读队列一个写队列
7.
8.          Monitor_reader_writer() //构造函数
9.          {
10.             reader_num = 0;
11.             writer = false
12.         }
13.
14.     public:
15.         void start_read()
16.         {
17.             if (writer) read_q.wait(); //如果有读者在读就等待
18.             reader_num ++;
```

```
            if (!write_q.queue && read_q.queue) read_q.signal(); //如果没有写者等着就唤醒读者
        }

        void end_read()
        {
            reader_num --;
            if (reader_num == 0 && write_q.queue) write_q.signal(); //如果没有读者了就唤醒写者
            else if (!write_q.queue  && read_q.queue) read_q.signal()(); //没有写者，继续唤醒读者
        }

        void start_write()
        {
            If (reader_num>0 || writer) write_q.wait(); //如果有读者在读或有写者在写，等待
            writer = true; //如果可以了就标明正在写
        }

        void end_write()
        {
            writer = false;
            if (write_q.queue) write_q.signal(); //如果有写者等优先写者
            else if (read_q.queue) read_q.signal(); //否则就唤醒读者
    } Mon
//----------以上是管程部分，接下来是读者和写者----------
void Read()
{
    while(1)
    {
        Mon.start_read();

        reading from memory

        Mon.end_read();
    }
}

void Write()
{
    while(1)
    {
        Mon.start_write();

        writing into memory

        Mon.end_write();
    }
}

void main()
{
    cobegin
     Read();
     Write();
    coend
}
```