تمرینات سری چهارم درس اصول سیستم ای عامل

استاد محمود فصلعلی

سروش حیدری - 96222031

سوال 1----------

روش های سمافور نسبت به بقیه روش های کنترل ناحیه بحرانی مزیت های زیر را داراست

- روش های سمافور (باینری و عددی) بصورت مستقل از ماشین روی زیر هسته ها پیاده سازی میشوند
- به دلیل دارا بودن خاصیت busy waiting روش سمافور زمان و منبع هدر رفته ای ندارد و زمان پردازشی غیر ضروری را هدر نمیدهد
- نوعی از روش سمافور وجود دارد (عددی) که اجازه میدهد برنامه ها به شکل صف گونه ای بتوانند برای ورود به ناحیه ی بحرانیشان منتظر بمانند این حالت میتواند مزایای خود را در شرایط مناسبش دارا باشد
- پیاده سازی این روش نسبت به بقیه روش ها راحت تر است و سهولت بیشتری دارد
- سمافور اجازه میدهد بیش از 1 نخ دسترسی به ناحیه بحرانی داشته باشد

سوال 2 ----------

```
BufferSize = n;   // size buffer
count = 0;    //tedad item haye toye buffer

 Producer(string item)          // code marboot be producer
 {
  int widget;
  WHILE (true) {

    IF(count==BufferSize)  {
          TestAndSet(&lock)  ;              // agar buffer por bashe lock ro set mikonim
          }

   While (TestAndSet(&lock) == 1)       // ta vaghti ke buffer por hastesh lock ro 1 negah midarim
       {
        IF (count!=BufferSize) {
              Lock = 0 ;                // vaghti buffer ma jaye khali dashte bashe lock ro 0 mikonim
       }
    }
   put_item(item);              // vaghti az loop biyaym biroon producer mitoone item jadid add kone
   count = count + 1;            // vaghti item jadid add she counter ro yedoone ezafe mikonim
}


 Consumer(item)             // code marboot be consumer
 {

  WHILE(true) {
   IF(count==0)
      TestAndSet(&lock) ;
      While (TestAndSet(&lock) == 1)       // ta vaghti ke buffer khali hastesh lock ro 1 negah midarim
       {
        IF (count != 0) {
              Lock = 0 ;                // vaghti buffer ma dige khali nabashe lock ro 0 mikonim
       }
    }
```

```
remove_item(item);        // vaghti ke lock 0 bashe customer mitoone item ro az buffer bardare
count = count - 1;        // count ro yeki kam mikonim
Consume_item(item);       // consumer mitoone itemi ke bardashte ro estefadde kone
```

سوال 3----------

برای حل این مشکل ناحیه بحرانی هم مشتری و هم تولید کننده را درون یک کلاس مونیتور پیاده سازی میکنیم به این شکل که هر کدام از این دو شخص که نیاز به ورود به ناحیه بحرانی داشتند تابع مورد نظر خود را از کلای مونیتور صدا میزنند و این توابع به شرح زیر است

تابع add : این تابع برای ورود تولید کننده به ناحیه بحرانی خود درون مونیتور پیاده سازی شده به این شکل که اگر بافر پر بود توسط wait() تولید کننده رو مجبور به انتظار میکنیم تا زمانی که بافر جای خالی پیدا کند و توسط سیگنالی که از سمت تابع remove() به طرف تولید کننده ارسال میشود اجازه دسترسی به ناحیه بحرانی اش و دسترسی به محصول را به او بدهد .همچنین اگر جای خالی وجود داشت و بعد از اضافه کردن محصول به بافر حال مقدار متغیر شمارنده ما 1 بود یعنی قبل از اضافه کردن محصول بافر خالی بوده در این صورت با یک سیگنال signal(consumer) مشتری را خبر کرده و به او اجازه دسترسی به ناحیه بحرانی را میدهیم

تابع remove : این تابع برای ورود مشتری به ناحیه بحرانی خود درون کلاس مونیتور پیاده سازی شده به این شکل که اگر بافر خالی باشد و محصولی برای برداشتن وجود نداشته باشد با استفاده از wait(consumer) مشتری را وادار به انتظار میکنیم تا وقتی که محصولی وارد بافر شود و توسط سیگنالی که تابع add به مشتری میدهد مشتری بتواند محصول را برداشت کند همچنین اگر بعد از برداشت محصول مقدار بافر برابر با n-1 باشد یعنی قبل از برداشت بافر پر بوده است در این صورت بعد از برداشت با استفاده از signal(producer) تولید کننده را صدا زده و به او اجازه اضافه کردن محصولاتش را میدهیم

کد زیر تماما از توضیحات بالا پیروی میکند و نیازی به کامنت های اضافی نیست .

```
Class Monitor {
        Condition full, empty;
        Int count;

        Function add() :
                If (count == n) { wait(producer) ; }
                put_item(item);
                count += 1;
                If (count ==  1) { signal(consumer) }
        Function remove() :
                If (count == 0) { wait(consumer) ; }
                remove_item(item);
                count -= 1;
                If (count ==  n-1) { signal(producer) }
        }

Function Producer( String item ) :
{
        While (true) {
                Make_item(item)
                Monitor.add() ;  }
}
Function Consumer( String item ) :
{
        While (true) {
                Monitor.remove()
                Consume_item ;  }
}
```