پاسخ نامه تمرین هشتم درس سیستم‌های عامل

استاد درس: دکتر زرندی

پاییز ۹۹

سوال ۱

یک راه برای پیاده سازی یک مانیتور به یک سمافور باینری (mutex) ، یک سمافور عادی و یک شمارنده نیاز دارد.

هر فرایند باید قبل از ورود به مانیتور wait(mutex) و بعد از خروج از آن signal(mutex) را صدا بزند. این کار انحصار متقابل در داخل مانیتور را تضمین می کند. (به این معنی که در لحظه فقط یک فرایند در داخل مانیتور در حال **اجرا** شدن است)

سمافور بعدی که اسمش را next می گذاریم برای کنترل و منتظر نگه داشتن فرایند هایی است که قبلا وارد مانیتور شده اند. در صورتی که یک فرایند داخل مانیتور به هر دلیلی (تست یک متغیر شرطی) نیاز به صبر داشته باشد ، با این سمافور نسبت به فرایند هایی که هنوز وارد مانیتور نشده اند اولویت می یابد. این موضوع در اسلاید بعدی که مربوط به متغیر شرطی است مشاهده می شود.

شمارنده ، که اسمش را next_count می گذاریم ، برای شمارش تعداد فرایند هایی است که در صف next قرار گرفته اند.

لذا پیاده سازی هر تابع دلخواه F به صورت زیر خواهد بود:

```
wait(mutex);
    ...
  body of F
    ...
if (next_count > 0)
  signal(next);
else
  signal(mutex);
```

سوال ۱

Wait:

```
x_count++;
if (next_count > 0)
   signal(next);
else
   signal(mutex);
wait(x_sem);
x_count--;
```

Signal:

```
if (x_count > 0) {
   next_count++;
   signal(x_sem);
   wait(next);
   next_count--;
}
```

همچنین لازم است هر متغیر شرطی مثل X را با یک سمافور و شمارنده مدل کنیم.

به این صورت که سمافور x_sem تعداد نمونه های موجود از متغیر شرطی و شمارنده تعداد فرایند هایی که منتظر x هستند را نگه داری می کند.

با این تفاسیر توابع wait و signal متغیر شرطی به صورت مقابل خواهد بود:

دقت کنید که در تابع wait ابتدا برسی می شود تا نوبت را به فرایند هایی بدهد که در داخل مانیتور هستند و اگر موردی نبود (next_count < 0) نوبت را به فرایند های خارج از مانیتور می دهد.

همچنین در تابع signal تنها در صورتی که فرایندی داخل تابع wait مانده باشد به آن اجازه اجرا می دهد. و همچنین چون داخل مانیتور هستیم روی next منتظر می ماند.

سوال ۲

برای هر صف یک سمافور باینری در نظر می گیریم (q1, q2) که تعیین می کند الان نوبت کدام صف است. مقدار دهی اولیه آنها باید طوری باشد که یکی صفر و دیگری ۱ باشد. (فرقی نمی کند کدام یک) این موضوع برای رعایت شرط نوبت دهی صف ها است.

همچنین دو شمارنده برای مشخص کردن تعداد افراد هر صف در نظر می گیریم (q1count, q2count) برای برسی خالی یا پر بودن صف ها. (تا اگر صفی خالی بود ، صف دیگر منتظر نماند)

در نهایت از یک سمافور باینری mutex برای اطمینان از انحصار متقابل در مقابل نانوا کمک می گیریم.

سوال ۲

```
Semaphore q1,q2 = (0,1)
Semaphore mutex = 1
int q1count, q2count = 0
```

Queue 1:

```
Get_bread() {

        q1count ++;
        if (q2count > 0)
                wait(q1);

        wait(mutex)
        … take the bread …
        signal(mutex)

        q1count --;
        if (q2count > 0)
                signal(q2);

        }
```

Queue 2:

```
Get_bread() {

        q2count ++;
        if (q1count > 0)
                wait(q2);

        wait(mutex)
        … take the bread …
        signal(mutex)

        q2count --;
        if (q1count > 0)
                signal(q1);

        }
```

سوال ۳

```
Void BeginRead()
{
        if (Active_Writers == 1 || WaitingWriters > 0)
        {
                Waiting_Readers += 1;
                Wait(Read);
                Waiting_Readers –= 1;
        }
        Active_Readers += 1;
        Signal(Read);
}
```

```
Void BeginWrite()
{
        if (Active_Writers == 1 || Active_Readers > 0)
        {
                Waiting_Writers += 1;
                wait(Write);
                Waiting_Writers –= 1;
        }
        Active_Writers = 1;
}
```

```
Void EndRead()
{
   Active_Readers - = 1
   // once a reader is done , it checks for waiting writers
   if (Active_Readers == 0) // (only if it was the last reader)
                Signal(Write);
}
```

```
Void EndWrite()
{
   Active_Writers = 0;
   // Check if any readers are waiting
   if (Waiting_Readers > 0)
                Signal(Read);
   else
                Signal(Write);
}
```

سوال ۳

بله – به این دلیل که در تابع begin read هیچ شرطی برای جلوگیری از ورود چند خواننده نداریم و تنها محدودیت بر سر نویسنده می باشد.

برتری های پیاده سازی با monitor در وهله اول کاهش قحطی زدگی و افزایش انصاف در بین خواننده ها و نویسنده ها ، در عین رعایت سادگی و قابل فهم بودن آن است. در این قطعه کد سعی شده تا اگر خواننده ای مشغول است ، نویسنده ها منتظر بمانند ، و اگر نویسنده ای مشغول است ، خواننده ها منتظر بمانند ، و در اولین فرصت به همه خواننده ها فرصت ورود می دهد.

همچنین عدم برخورد با مشکلاتی مثل بن‌بست نیز هم جزو برتری های این پیاده سازی می باشد.