به نام خدا

سیستم های عامل (بهار ۱۴۰۱)

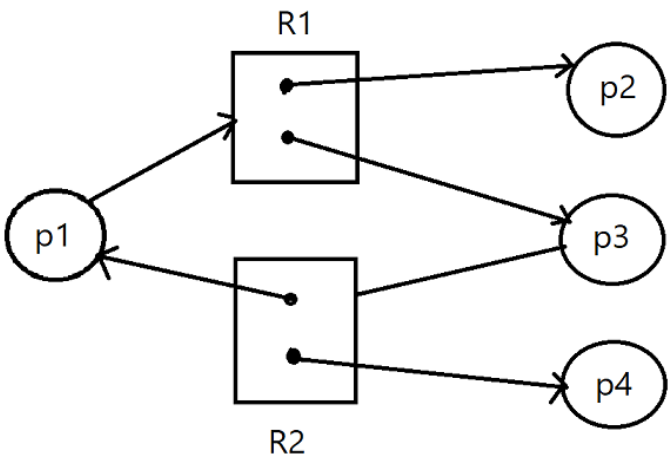
پاسخنامه تمرین چهارم

1) الف) فرض کنید در مجموع 9 واحد از یک نوع منبع موجود است، با توجه به اطلاعات داده شده راجع به هر پردازه ، نشان دهید که آیا دنباله ای وجود دارد که در حالت safe state باشد؟

به عنوان مثال دنباله ی (P4, P2, P3, P1) ،از چپ به راست، در حالت ایمن نیست.

process	used	Max
p1	2	7
p2	1	6
p3	2	5
p4	1	4

ب) آیا در گراف زیر دور(cycle) وجود دارد؟ deadlock چطور؟ علت را بیان کنید.



پاسخ:

الف)

Process	Used	Max	Need
P1	2	7	5
P2	1	6	5
P3	2	5	3
P4	1	4	3

Currently Available resources = Available - Allocated resources = 9 - 6 = 3

اگر درخواست P4 ابتدا پذیرفته شود، پس از اجرا حداکثر 4 منبع را آزاد می کند و اگر P1 یا P2 بعدی تخصیص داده شود، درخواست آنها قابل انجام نیست. زیرا هر دو به 5 منبع نیاز دارند.

(P3, P1, P2, P4)پاسخ ممکن

ب)در این گراف دور وجود دارد ولی بن بست نداریم. زیرا منابعی وجود دارند که در داخل دور نیستند و در صورت رها شدن توسط پردازه های 2 و 4 میتوان از بن بست خارج شد.

2) سیستم زیر را در نظر بگیرید ، آیا سیستم در حالت امن است؟به طور کامل توضیح دهید.

ستون available بعد از اختصاص دادن منابع به پردازه ها است.

process	MAX A B C D	Allocation A B C D	Available A B C D
p0	2 1 0 6	1 0 0 4	1 1 2 3
P1	0 5 7 1	0 0 1 1	
P2	6 5 2 3	4 5 2 1	
P3	3 5 6 1	3 3 6 0	
P4	6 5 6 1	2 1 2 0	

پاسخ:از الگوریتم بانکدار استفاده میکنیم.

1) Let Work and Finish be vectors of length 'm' and 'n' respectively.

Initialize: Work = Available

Finish[i] = false; for i=1, 2, 3, 4….n

2) Find an i such that both

a) Finish[i] = false

b) Need$_i$ <= Work

if no such i exists goto step (4)

3) Work = Work + Allocation[i]

Finish[i] = true

goto step (2)

4) if Finish [i] = true for all i

then the system is in a safe state

در ابتدا مقادیر موجود در آرایهٔ Work برابر است با [3, 2, 1, 1]

و آرایهٔ Finish به ازای تمام پردازه ها مقدار false دارد [false, false, false, false, false]

در میان پردازه ها یکی از پردازه هایی که شروط 2 را داشته باشد انتخاب میکنیم.(مثلا P0)

پس از اجرای مرحلهٔ 3 Work برابر با [7, 2, 1, 2]است و آرایهٔ Finish برابر است با

[true, false, false, false, false]

سپس پردازه ی P2 را انتخاب میکنیم. پس از اجرای مرحلهٔ 3، Work برابر است با [8, 4, 6, 6]

و finish برابر با [true, false, true, false, false] یشود.

سپس پردازهٔ P3 را انتخاب میکنیم. پس از اجرای مرحلهٔ 3 ، Work برابر است با [8, 10, 9, 9]

و finish برابر با [true, false, true, true, false]

در انتها آرایهٔ Finish به ازای تمام پردازهها برابر با true میشوددر نتیجه میتوان به درخواست های پردازه ها پاسخ داد. در نتیجه در حالت امن قرار داریم.

3) با توجه به الگوریتم های زیر، برای هر کدام مشخص کنید که کدام یک از شروط انحصار متقابل، پیشرفت و انتظار محدود را رعایت می کنند و کدام یک را نقض می کنند. دلایل خود را بنویسید.

الف)

```
Method used by P1
while(true){
    while(turn!=1);
    critical section
    turn = 2;
outside of critical section
}

Method used by P2
while(true){
    while(turn!=2);
    critical section
    turn = 1;
outside of critical section
}
```

ب)

```
do {
        flag[j] = true;
        turn = j;
        while ( flag[i] && turn == j);
            critical section
        flag[j] = false;
            remainder section
} while (true);
```

پاسخ:

الف)

انحصار متقابل:

وجود دارد، زیرا مقدار turn نمی تواند همزمان ۱ و ۲ باشد.

پیشرفت:

وجود ندارد. فرض کنید یک پردازه کار خود را انجام داده و وارد بخش non-critical section شود. اگر این بخش خیلی سریع تمام شود، نوبت از آن جایی که دست پردازه دیگر است، نمی تواند وارد شود.

انتظار محدود:

وجود دارد. زیرا نوبت بین این در پردازه به صورت یکی در میان جابجا می شود.

ب)

انحصار متقابل:

وجود دارد، فرآیندی که نوبت را به فرآیند دیگر بدهد خود نخواهد توانست وارد بخش بحرانی شود تا هنگامی که فرآیند دیگر از بخش بحرانی خارج شود .

پیشرفت:

وجود دارد، زیرا حالتی وجود ندارد که هر دو فرآیند قادر به عبور از حلقه while نباشند.

انتظار محدود:

وجود دارد، چون به دلیل تعارف نوبت امکان ندارد که یک فرآیند برای همیشه پشت حلقه while باقی بماند.

راه حل ساده تر:

این الگوریتم همان الگوریتم پترسون است زیرا تنها اندیس های i و j در همه کاربردهای متغیر flag باهم جا به جا شده اند . پس همه شروط برقرار هستند.

4) قطعه کد زیر را در نظر بگیرید. در این تابع قصد داریم عملیات ضرب بین مقدار op1 و مقداری که P_op2 به آن اشاره دارد را انجام دهیم. حاصل ضرب باید در حافظه ای که P_op2 به آن اشاره می کند ذخیره شود. این تابع را طوری با دستور compare_and_swap کامل کنید که عملیات ضرب به صورت اتمی انجام شود. همچنین برقرار بودن یا نبودن هر کدام از شروط سه گانه را با دلیل شرح دهید.

```
int multiplication(int op1, int *P_op2){
        // TODO
    return *P_op2, *op1;
}
```

پیاده سازی تابع compare_and_swap را به صورت زیر در نظر بگیرید:

```
bool compare_and_swap (int *value, int old, int new){
        if(*value != old){
    return false;
        }
    *value = new;
      return true;
}
```

پاسخ:

```
Int multiplication(int op1, int *P_op2) {

    While(!compare_and_swap(P_op2, *P_op2, *P_op2 * op1));

}
```

با توجه به اینکه هر فرآیند که مقدار حاصل حاضر P_op2 را خوانده است، در صورتی که به هنگام تغییر مشاهده کند که مقدار P_op2 با آن چیزی که از قبل خوانده تفاوت دارد return false می کند و در حلقه باقی می ماند. در نتیجه انحصار متقابل داریم.

زمانی که یک فرآیند تغییرات خود را اعمال کند و کارش را به پایان برساند؛ فرآیندهایی که در حلقه منتظر مانده اند، تابع compare_and_swap را صدا می زنند و با توجه به اینکه هریک از آنها مقدار به روز شده را می خوانند، می توانند تغییرات خود را اعمال کنند؛ بدین ترتیب پس از پایان کار فرآیند قبل، یک فرآیند جدید می تواند تغییرات خود را اعمال کند، در نتیجه پیشرفت داریم.

به دلیل اینکه هیچ کنترلی روی فرآیندهای در حال اجرا نداریم، ممکن است یک فرآیند همیشه در حلقه While گیر کند و هیچگاه نتواند عملیات را به اتمام برساند. در نتیجه این متود هیچ تضمینی به انتظار محدود نمی دهد.

5) انتظار مشغول (waiting busy)چیست؟ سایر انتظارهاي موجود در سیستم عامل کدامند؟ آیا به صورت کلی میتوان از انتظار مشغول اجتناب کرد؟ پاسخ خود را توضیح دهید.

پاسخ:

انتظار مشغول تکنیکی است که در آن یک فرایند، مکرراً یک شرط خاص را بررسی میکند تا ببیند آیا آن شرط برقرار است یا خیر؛ مثال یک حلقه while بدون اینکه قطعه کدی را اجرا کند، فقط در انتظار باطل شدن شرط تکرار برای خروج است. وجود این نوع انتظار باعث میشود که یک پردازه بدون اینکه کار مفیدی انجام دهد، منابع اجرایی سیستم را در اختیار داشته باشد و در ظاهر مشغول به کار باشد، در حالی که کاری انجام نمیشود.

علاوه بر انتظار مشغول، انتظار محدود (Bounded Wait)،انتظار چرخش (Spin Wait) ،انتظار مسدود شده(Blocked Wait)و انتظارهایی مانند انتظار برای I/O و انتظار پردازه برای شروع به اجرا (تغییر وضعیت از ready به run) وجود دارد.

مشکل انتظار مشغول از طریق استفاده از Sleep یا Block کردن یک پردازه قابل جلوگیری است، البته این روشها نیز برای پردازه سربار خودشان را دارند.

موفق باشید

تیم تدریس یاری درس سیستم های عامل