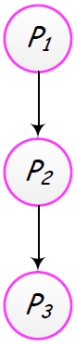
باسمه تعالی

دانشگاه صنعتی اصفهان
سیستم های عامل ـ تمرین سوم
موعد تحویل: جمعه ۵ دی ۹۹

سوالات تئوری

سوال ۱ (الف ۳۰ نمره ـ ب ۳۰ نمره)

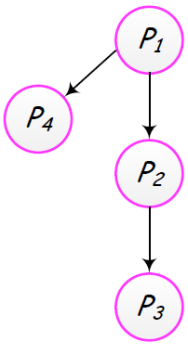
در اغلب سیستم های مدرن تعدادی از فرایندها و نخ ها [۱] که بعضاً گروه هایی از آنها به عنوان مؤلفه های یک سیستم نرم افزاری با یکدیگر همکاری دارند، به صورت همروند [۲] اجرا می شوند. فرایندهای همروند برای انجام وظیفه خود مکرراً در حال ارتباط با یکدیگر هستند. در چنین پروسه ای گاهی فرایندها نیاز به نتیجه فرایند دیگری برای انجام کار خود دارند و گاهی منابع سیستم نیز به صورت مشترک بین آنها استفاده می شود. زمانی که منبعی بین چندین فرایند بصورت مشترک در حال استفاده است، امکان ایجاد شرایط رقابتی برای در اختیار گرفتن آن نیز مطرح می شود. در چنین شرایطی باید توجه داشت که علاوه بر رابطه ای که از لحاظ روند اجرایی بین فرایندها برقرار است، در استفاده از منبع مشترک نیز بن بست رخ ندهد. به طور مثال در گراف زیر سه فرایند نشان داده شده است.



در این گراف فرایند P1 برای اجرا نیاز به نتیجه فرایند P2 دارد و فرایند P3 نیز به همین ترتیب باید بعد از اتمام کار فرایند P2 شروع به کار نماید. بنابراین اگر منبعی در اختیار فرایند P1 قرار دارد، بعد از اتمام کارش آزاد شده و می تواند در اختیار فرایند دیگری که به آن نیاز دارد (در اینجا یعنی فرایند P2) قرار گیرد. در این حالت شرایط رقابتی برای منبع پیش نمی آید. اما فرض کنید که گراف همروندی به صورت زیر باشد:

---

[۱] Threads
[۲] Concurrent



در این حالت فرایند P4 در سطح یکسانی از ارتباط با فرایند P2 قرار دارد. بدین معنا که بعد از اتمام کار P1 منبع اشتراکی به صورت مشترک مورد استفاده برای کار این دو فرایند قرار می گیرد. حال اگر قرار بر جلوگیری از وقوع بن بست در سیستم باشد، نیاز است که مقدار منبع موجود مطابق با رابطه زیر باشد:

$\sum_{i=1}^{n} Max[i] < n + E$

به طوری که مقدار Max[i] حداکثر نیاز فرایند i به منبع مورد نظر بوده و E کل موجودی منبع است. مقدار n در این رابطه تعداد فرایندها را نشان می دهد. این رابطه باید در هر سطح از گراف همروندی و بین فرایندهای آن سطح برقرار باشد. به طور مثال برای گراف فوق باید شرایط زیر در استفاده از منبع مشترک برقرار باشد تا احتمال بن بست صفر باشد:

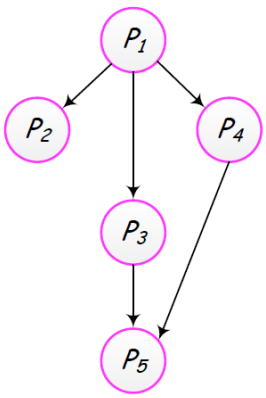
Max[1]<1+E

Max[2]+Max[4]<2+E

Max[3]<1+E

الف) حال فرض کنید گراف مربوط به رابطه همروندی بین ۵ فرایند به صورت زیر است که همه این فرایندها نیاز به یک منبع مشترک R1 واحد برای اجرا دارند. اگر حداکثر نیاز هر یک از آنها نیز مطابق جدول نشان داده شده باشد، محاسبه کنید که حداقل چند نسخه از این منبع نیاز است تا احتمال بن بست صفر باشد.

	P_1	P_2	P_3	P_4	P_5
حداکثر نیاز فرایند به منبع R_1	3	3	2	2	8



ب) (اختیاری) رابطه ارائه شده در این سوال را به حالتی که بجای یک منبع از چندین منبع در سیستم به صورت مشترک بین فرایندها استفاده می شود، بدست آورید و عملکرد آن را به طور مختصر توجیه کنید.

سوال ۲ (۳۰ نمره)

در سوال قبل یک روش محافظه کارانه برای جلوگیری از بن بست بکار گرفته شد که می تواند گاهی منجر به اتلاف شدیدی برای منابع گردد. روش دیگری که در اجتناب از وقوع بن بست بکار گرفته می شود که گرچه به بن بست نزدیکتر است اما با عدم پذیرش درخواست هایی که ریسک بن بست دارند و باعث ایجاد حالت نا امن می شوند، سعی در جلوگیری از بن بست دارد.

در این روش، یک حالت در صورتی امن [۳] نامیده می شود که بتوان یک توالی از حالت های بعدی را چنان پیدا کرد که همه فرایندها یکی یکی مقدار منابع مورد نیاز خود را در اختیار گرفته و خاتمه یابند. به عبارت دیگر بتوان یک فرایند را پیدا کرد که تمام منابع مورد نیاز خود را بگیرد و بعد از اجرای کامل، آن منابع را آزاد نماید. سپس با توجه به منابع آزاد شده جدید، یک فرایند دیگر پیدا کرد که بتواند تا آخر اجرا شود و به همین ترتیب تا آخرین فرایند پیش رفت. به این توالی یک مسیر امن گفته می شود و راه فراری از بن بست احتمالی محسوب می گردد.

برای استفاده از چنین روشی ابتدا باید بررسی کرد که هر فرایند چه میزان منبع برای تکمیل کار خود نیاز دارد. حال باید تصمیم گرفت که کدام فرایند قادر است با توجه به منابع موجود و آزاد سیستم اجرا شود و کار خود را خاتمه دهد. یافتن مسیری که دارای چنین روند و شرایطی باشد می تواند یک مسیر امن برای اجرای فرایندهای سیستم تلقی گردد.

حال سیستمی را فرض کنید که شامل ۵ فرایند باشد به گونه ای که از دو منبع مختلف در آن استفاده می شود که در جدول زیر نشان داده شده است. محاسبه کنید که برای امن بودن سیستم حداقل به چه میزان از منبع R1 نیاز است؟ پاسخ خود را با دلیل به طور مختصر توجیه کنید.

فرایند	منابع اختصاص یافته به فرایند		حداکثر منابع مورد نیاز فرایند	
	R_1	R_2	R_1	R_2
P_1	0	2	5	2
P_2	2	5	2	10
P_3	4	0	4	5
P_4	1	1	1	4
P_5	0	0	9	5

تعداد منابع باقی مانده	
R_1	R_2
x	3

سوال ۳ (الف ۳۰ نمره_ب ۳۰ نمره)

الف) اگر i و j متغیرهای محلی و S متغیر مشترک با مقدار اولیه ۴ فرض شود، توضیح دهید که با بررسی حالات ممکن وقوع وقفه در روال اجرای برنامه، پس از اجرای کامل فرایندهای P1 و P2 (به صورت تصادفی) در یک سیستم اشتراک زمانی، حداکثر مقدار S چه مقداری می تواند داشته باشد؟ (روند دستیابی به این مقدار را توضیح دهید)

---

[۳] Safe

```
P1: for i=1 to 2 do
    S=S*3;

P2: for i=1 to 2 do
    S=S/2;
```

ب) اگر i و j متغیرهای محلی و S متغیر مشترک با مقدار اولیه ۲ فرض شود، توضیح دهید که با بررسی حالات ممکن وقوع وقفه در روال اجرای برنامه، پس از اجرای کامل فرایندهای P1 و P2 (به صورت تصادفی) در یک سیستم اشتراک زمانی، حداقل مقدار S چه مقداری می تواند داشته باشد؟ (روند دستیابی به این مقدار را توضیح دهید)

```
P1: for i=1 to 2 do
    S=S*2;

P2: for i=1 to 2 do
    S=S+2;
```

برای پاسخ به این سوال باید توجه داشته باشید که دستوری مانند S=S*3 به زبان اسمبلی معادل با سه دستور زیر است:

$$MOVE\ REG,\ S$$
$$MUL\ REG,\ \#3$$
$$MOVE\ S,\ REG$$

سوال ۴ (الف ۳۰ نمره ـ ب ۳۰ نمره)

دو فرایند P0 و P1 با ساختار زیر، توابع معرفی شده در این سوال را بکار می برند.

```
P(int i){
    while(TRUE){
        enter-region();
        critical-region();
        leave-region();
        non-critical-region();
    }
}
```

فرض کنید که برای مسأله انحصار متقابل [۴] دو راه حل قسمت های الف و ب پیشنهاد شده است. توضیح دهید که هر یک از این راه حل ها دارای چه مشکلی هستند؟ (دلیل ادعای خود را توضیح دهید.)

[۴] Mutual Exclusion

الف)

```
bool n[2]={false, false}
int turn=0;

void enter-region(int id){
    n[id]=true;
    while(turn!=id) {
        while(n[1-id]);
        turn=id;
    }
}

void leave-region(int id){
    n[id]=false;
}
```

ب)

```
bool n[2]={false, false}
int turn=0;

void enter-region(int id){
    n[id]=true;
    while(turn!=id) {
        while(n[1-id])
            turn=id;
    }
}

void leave-region(int id){
    n[id]=false;
}
```

سوال ۵ (۳۰ نمره)

یکی از روش های همگام سازی [۵] استفاده از مفهومی به نام مانع [۶] است. گاهی یک برنامه طی فازهای مختلفی انجام می شود و هر نخ پس از اتمام فاز جاری و برای ورود به فاز بعدی، باید منتظر تکمیل فاز جاری تمامی نخ های دیگر باشد. این مسأله از آن جهت است که تمامی نخ ها در یک سیستم کار یکسانی را انجام نمی دهند. این امکان بسیار محتمل است که نخی کار خود را زودتر به اتمام برساند درحالی که نخ دیگری به دلیل کار طولانی تری که بر عهده دارد به زمان بیشتری برای تکمیل کار خود نیاز پیدا می کند. به این ترتیب تمامی نخ ها با اتمام

[۵] Synchronization
[۶] Barrier

فاز اول کار خود پشت مانع صبر می کنند و منتظر اتمام کار سایر نخ ها می شوند. با فراهم شدن این شرایط، امکان ادامه اجرای نخ ها برای فاز بعدی فراهم می شود. بنابراین ورود به فاز بعدی به صورت همزمان توسط نخ ها انجام خواهد شد. برای این کار می توان از اجرای Barrier-done در انتهای هر فاز اجرای هر نخ استفاده کرد.

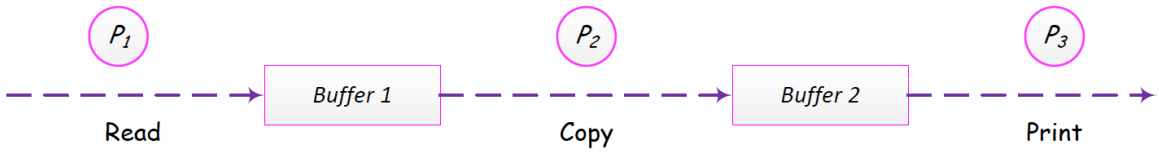
فرض کنید به تعداد N نخ داریم که در سیستمی به شیوه توضیح داده شده کار می کنند. توضیح دهید که چرا بکارگیری کد زیر در این رابطه ممکن است عملکرد درستی نداشته باشد.

```
semaphore mutex = 1;
semaphore barrier = 0;
int count = 0;


void barrier-done() {
    wait(mutex);
    count++;
    if (count < N ) {
        post(mutex);
        wait(barrier);
    }
    else {
        post(mutex);
        count = 0;
        for (int i = 1; i < N; i++) {
            post(barrier);
        }
    }
}
```

سوال ۶ (۳۰ نمره)

فرض کنید در عملیات چاپ یک فایل، سه فرایند درگیر باشند. فرایند A محتوای فایل را از دیسک می خواند و درون بافر اول می ریزد. فرایند B اطلاعات را از بافر اول به بافر دوم کپی می کند. در نهایت نیز فرایند C فایل را از بافر دوم برداشته و محتوای آن را چاپ می کند.



فرض کنید که هر سه فرایند در هر لحظه روی یک فایل کار می کنند و ظرفیت هر دو بافر نیز به اندازه فایل باشد.

الف) مشخص کنید که دربخش های مشخص شده در کد زیر چه دستوراتی باید قرار گیرد.

ب) در صورتی که بخواهیم از condition variable استفاده کنیم، راه حل پیشنهادی شما برای فرایند A به

چه صورت خواهد بود. در واقع سعی کنید دستورات قسمت الف را برای فرایند A با این امکان تعیین کنید.

راهنمایی: برای پر کردن بخش خواسته شده از کد از دستورات wait، signal، post، copy و print استفاده کنید.

```
semaphore empty1 = 1;
semaphore empty2 = 1;
semaphore full1 = 0;
semaphore full2 = 0;

Process-A () {
    while(1) {

    yourcode

    }
}

Process-B () {
    while(1) {

    yourcode

    }
}

Process-C () {
    while(1) {

    yourcode

    }
}
```

سوال ۷ (۳۰ نمره)

توضیح دهید مشکل پیاده سازی زیر از الگوریتم پترسون چیست؟ توجه کنید که i می تواند دو مقدار صفر و یک را داشته باشد. و این الگوریتم بین دو فرایند P0 و P1 اجرا می شود.

```
Process Pi:
    do {
        flag[i]=TRUE;
        turn=i;
        while(flag[j]&&turn==j);
        critical-section();
        flag[i]=FALSE;
        non-critical-section();
    } while(TRUE)
}
```

سوال ۸ (۳۰ نمره)

فردی الگوریتم همگام سازی زیر را پیشنهاد داده است. ادعا می شود این الگوریتم کارکرد درستی دارد. صحت این ادعا را با دلیل یا مثال نقض بررسی کنید.

```
while (true) {
    while (lock != 0);
    lock = 1;
    critical_section();
    lock = 0;
    non_critical_section();
}
```

سوالات برنامه نویسی

سوال ۹ (الف ۳۰ نمره ـ ب ۶۰ نمره)

کد Semaphore.c که در پیوست تکلیف قرار دارد، دارای مشکل بن بست است.

الف) این کد را تحلیل کنید و سناریویی را که طی آن بن بست می تواند اتفاق بیفتد، ارائه دهید.

ب) این کد را طوری اصلاح نمایید که مشکل برطرف شود و روند آن را نیز توضیح دهید.

راهنمایی: به مسأله استفاده از چند سمافر در این کد توجه کنید.

سوال ۱۰ (الف ۳۰ نمره ـ ب ۶۰ نمره)

کد counter.c در پیوست تکلیف را در نظر بگیرید.

الف) این کد را اجرا نموده و خروجی آن را مشاهده کنید. عملکرد این برنامه را توضیح دهید.

ب) این کد را به گونه ای اصلاح نمایید که مقدار خطای چاپ شده در خروجی صفر باشد.

راهنمایی: برای این کار نیاز است که از سمافر برای دسترسی انحصاری به متغیرهای مشترک [۷] استفاده نمایید.

سوال ۱۱ ـ اختیاری (الف ۲۰ نمره ـ ب ۱۰۰ نمره)

در این سؤال برای mutex دو عملیات release و acquire و برای سمافور، دو عملیات post و wait تعریف شده است.

الف) تفاوت semaphore و mutex چیست؟ آیا اگر در یک زبان برنامه نویسی فقط بکارگیری از semaphore در نظر گرفته شده باشد، این زبان در حل مسائل همگام سازی به مشکل می خورد؟ اگر بله توضیح دهید در چه صورت محدودیت ایجاد می شود و اگر خیر توضیح دهید چگونه سمافور می تواند عملکرد mutex را نیز داشته باشد.

ب) فرض کنید یک بانک دارای n حساب مشتری است. در توابع transfer ، deposit و withdraw که قسمت بحرانی مشخص شده است، با تعریف و بکار بردن سمافور یا mutex های مناسب از پیش آمدن مشکل همزمانی تغییر دو حساب جلوگیری کنید. همچنین هر یک از این عملیات یک تراکنش فرض می شود. بانک به دلیلی می خواهد از اجرای بیش از m تراکنش همزمان جلوگیری کند. پس در ارائه راه حل خود دقت کنید که انحصار متقابل رعایت شود، از بن بست جلوگیری شود و بیش از m تراکنش همزمان اجرا نشود. ضمنا اگر همزمان دو تراکنش واریز و برداشت بخواهد اجرا شود، تراکنش واریز باید اولویت داشته باشد. بقیه تراکنش ها اولویتی نسبت به هم ندارند.

Transfer(i،j){	withdraw(i،j){	deposit(i،j){
..	..	..
..	..	..
انتقال k ریال از حساب i به حساب j	برداشت k ریال از حساب i	واریز k ریال به حساب i
..	..	..
}	}	}

سوال ۱۲ ـ اختیاری (۱۰۰ نمره)

می خواهیم گذر ماشین ها از روی یک پل یک طرفه را مدیریت کنیم. این امر به صورتی است که تا وقتی ماشین هایی در حال عبور از پل از یک سمت هستند، به ماشین های دیگر که از سمت دیگر قصد ورود دارند اجازه ورود داده نشود. محدودیتی روی تعداد ماشین هایی که روی پل هستند وجود ندارد، اما می خواهیم مسأله گرسنگی را تا حدی حل کنیم. بدین منظور اگر ۵ ماشین از سمت شمال از پل رد شدند و تعدادی (یک یا بیشتر) ماشین در سمت جنوب قصد ورود به پل را داشتند، از ادامه عبور ماشین ها از سمت شمال جلوگیری کرده تا ماشین های مربوط به سمت جنوب از پل رد شوند. همین قانون برای سمت جنوب هم به صورت برعکس برقرار است.

با تابع north ماشینی که در شمال پل است، قصد ورود به پل را کرده و از پل رد می شود. همچنین با تابع south ماشین های سمت جنوب از پل رد می شوند.

سمافور یا Mutex های موردنیاز برای حل مسأله را تعریف کرده و دو تابع نامبرده را با استفاده از آن ها پیاده سازی کنید .

---

[۷] Shared variables

شبیه سازی

سوال ۱۳ ـ اختیاری (۶۰ نمره)

در کتاب اصلی درس (OSTEP[8]) در انتهای بیشتر بخش ها تمرین هایی در قالب کار با مجموعه ای از شبیه سازها آمده است. برای اطلاعات بیشتر در رابطه با این شبیه سازها می توانید به لینک زیر مراجعه کنید.
Homework
در این تمرین شما باید سوال ۸ و ۹ تکلیف در بخش بیست و هشتم کتاب (ch28:locks) را انجام دهید و با گرفتن عکس به صورت مرحله به مرحله از خروجی های شبیه ساز و توضیح در مورد جواب مساله (که توسط خود شبیه ساز داده می شود) در مورد تجربه خود در کار کردن با شبیه ساز گزارش دهید.

شیوه تحویل

برای این تمرین می بایست یک فلدر به نام studentid_hw3 بسازید (به جای studentid باید شماره دانشجویی خود را قرار دهید) که شامل دو فایل زیر باشد:

۱. یک فایل pdf: شامل پاسخ به تمام سوالات (ترجیحا به زبان فارسی) که می بایست با استفاده از LaTeX ایجاد شده باشد.

۲. یک فایل C: برای سوالات ۹ و ۱۰ و در صورت انجام سوالات ۱۱ و ۱۲، علاوه بر توضیحی که در فایل pdf فوق در مورد این سوال داده می شود باید فایل source code به زبان C نیز در این فلدر قرار داده شود.

سپس فلدر خود را با دستور زیر بایگانی و فشرده سازی کنید.

```
tar zcf studentid_hw3.tgz studentid_hw3
```

و تنها فایل studentid_hw3.tgz را در سامانه یکتا در قسمت مربوط به تکلیف سوم بارگذاری کنید.

موفق باشید

[8] Operating system: three easy pieces