



دانشکده مهندسی کامپیوتر

درس سیستم‌های عامل

---

تمرین سری چهارم


---

مدرسین ..... دکتر رضا انتظاری ملکی، دکتر وحید ازهری

تیم طراح ..... محمدحسین عباسپور – محسن رحیمی

تاریخ انتشار ..... ۱۴۰۲/۰۹/۲۰

تاریخ تحویل ..... ۱۴۰۲/۱۰/۰۱

در رابطه با تمرین 

➤ این تمرین شامل مبحث:

### • Synchronization

می باشد.

➤ نمره این تمرین از ۱۰۰ می باشد و بارم هر سوال روبه روی آن نوشته شده است.

➤ به هیچ وجه تمرینی را از دیگران کپی نکنید. در صورت مشاهده تقلب و کپی در تمرینات، نمره هر دو طرف صفر در نظر گرفته می شود.

## ۱- برداشت و واریز (۲۵ نمره)

در اینجا قصد داریم ناسازگاری و ناهماهنگی‌ای که ممکن است هنگام واریز و برداشت وجه از یک حساب بانکی رخ دهد را با mutex کنترل کنیم.

هر کس یک سرمایه‌ای برای خود دارد و در صورتی که هنگام برداشت مقدار موجودی برای آن شخص ناکافی باشد، می‌تواند از safe box که برای تمام اعضا مشترک است، استفاده کند. ولی هنگام واریز وجه به حسابش باید ابتدا بدهی خود را با safe box صاف کند و یا تا حد امکان بدهی خود را کاهش دهد. به عنوان مثال:

موجودی safe box برابر 1000\$ و موجودی شخص A برابر 200\$ می‌باشد.

۱. برداشت 450\$ از حساب شخص A: موجودی ناکافی است پس 200\$ از حساب خود شخص و 250\$ از safe box برمی‌دارد. در نتیجه موجودی شخص 250\$- می‌باشد. (یعنی 250\$ به safe box بدهکار است)
۲. واریز 600\$ به حساب شخص A: ابتدا 250\$ بدهی را با safe box صاف می‌کند و مقدار باقی مانده را به حساب خود واریز می‌کند. در نتیجه موجودی حساب شخص A برابر 350\$ می‌باشد.

هر تراکنش توسط یک thread انجام می‌شود. تراکنش یا واریز (deposit) است و یا برداشت (withdraw).

یکی از مشکلاتی که ممکن است رخ دهد هنگام برداشت همزمان از یک حساب است. به عنوان مثال موجودی حساب شخص A برابر 200\$ است و دو تراکنش (که توسط دو thread انجام می‌شوند)، یکی با مقدار 150\$ و دیگری با مقدار 100\$ بخواهند از این حساب پول برداشت کنند. اگر این نوع تراکنش را handle نکنیم ممکن است نتیجه نهایی این دو تراکنش اشتباه باشد. (آن عملیاتی که دیرتر رخ میدهد در اصل 50\$ را باید از safe box بردارد) همین مشکل ممکن است برای safe box رخ دهد.

در اینجا شما باید با کامل کردن قطعه کدی که در اختیار شماست، این مشکل را برطرف کنید.

برای این سوال یک گزارش از کدهای نوشته شده به همراه ورودی و خروجی و یک Makefile اضافه کنید.

▪ به کامنت‌ها توجه کنید

**۲- A-F-C (۲۵ نمره)**

ترتیب نوشتن در پراسس p1 به این شکل است:

F  
E  
G

ترتیب نوشتن در پراسس p2 به این شکل است:

A  
C  
B

با استفاده از semaphore این Process ها را به گونه‌ای کنترل کنید که A قبل از F و F قبل از C در خروجی چاپ شود.

برای این سوال یک گزارش از کدهای نوشته شده به همراه ورودی و خروجی و یک Makefile اضافه کنید.

### ۳- عملیات کپی (۲۵ نمره)

در این سوال باید عملیات کپی کردن فایل را پیاده سازی کنید.

شیوه حل باید به این صورت باشد که دو **thread** داریم که یکی فایل را از مبدا می خواند و در یک بافر (یک آرایه ساده) می نویسد. در ادامه یک **thread** دیگر این دیتا را می خواند و در فایل مقصد می نویسد. وقتی یک تکه از دیتا در فایل مقصد نوشته شد، فرستنده در بافر قسمت بعدی را می نویسد.

این سوال یک مثال از مسئله **producer-consumer** است که **producer** از فایل مبدا می خواند و **consumer** در فایل مقصد می نویسد.

عملیات نوشتن و خواندن از **buffer** باید با **mutex** کنترل شود.

برای چک کردن اینکه فایل به درستی منتقل شده است از دستور زیر استفاده کنید:

```
md5sum <source_file> <dest_file>
```

خروجی این دستور هش فایل های مربوطه است که باید یکسان باشد.

تایپ فایل ارسالی می تواند تکست یا باینری باشد. (حالت کلی پیاده سازی شود)

شیوه اجرای برنامه باید به این صورت باشد:

```
./copy <source_file_path> <destination_file_path>
```

**source\_file\_path** و **destination\_file\_path** به صورت آرگومان های ورودی تابع **main** باید دریافت شود. **copy** هم نام فایل اجرایی است و چک کردن آن در ورودی معنی ندارد.

برای این سوال یک **readme** ساده در حد اینکه چطور کدتان ران شود و یک **Makefile** هم اضافه کنید.

## ۴- تخصیص منابع به Threadهای مختلف (۲۵ نمره)

در این مسئله باید برنامه‌ای بنویسید که منابع محدودی را به تعداد بیشتری thread اختصاص دهد.

در ابتدای برنامه ۵ تا resource داریم و ۱۰ تا thread. هر thread هم تعداد مشخصی task (که هر task زمان رندومی برای اجرا نیاز دارد) برای انجام دادن دارد.

هر کدام از این threadها در یک حلقه تا زمانی که همه تسک‌هایش را کامل کند تلاش می‌کند که یکی از resourceها را به خود اختصاص دهد.

اگر resourceی به thread رسید، عملیات خود را انجام می‌دهد. عملیات آن را با چند ثانیه استراحت شبیه سازی کنید. هر thread در هنگام انجام عملیات خود عبارت زیر را print می‌کند (فقط یک بار چاپ می‌کند و در ادامه استراحت می‌کند):

```
Thread <threadId> is performing work with resource <resourceId>
```

برای شبیه سازی resourceها از استراکچر زیر استفاده کنید:

```
typedef struct {
    int resources[NUMBER_OF_RESOURCES];
    sem_t availableResources;
    pthread_mutex_t poolMutex;
} ResourceManager;
```

آرایه resources شناسه threadهای اختصاص داده شده را ذخیره می‌کند. اگر منبعی تخصیص داده نشده باشد، مقدار آرایه در آن index یک عدد مشخص (مثلاً یک عدد منفی) است. برای مثال اگر ایندکس ۰ این آرایه ۳ باشد به این معنی است که ریسورس ۰ به thread با id ۳ اختصاص داده شده است.

در ابتدای تخصیص منبع به یک thread باید با سمافور چک شود که منبع در دسترس باشد وگرنه ترد منتظر می‌ماند (همزمان بیشتر از ۱۰ ریسورس نباید اختصاص داده شود).

در هنگام تخصیص آرایه resources باید این آرایه لاک شود که یک ایندکس آن به دو ترد اختصاص داده نشود.

در نهایت خروجی احتمالی برنامه شما به این صورت خواهد شد:

```
Thread 1 is performing work with resource 1
Thread 2 is performing work with resource 2
Thread 5 is performing work with resource 5
Thread 4 is performing work with resource 4
Thread 3 is performing work with resource 3
Thread 6 is performing work with resource 4
Thread 7 is performing work with resource 3
Thread 8 is performing work with resource 4
Thread 9 is performing work with resource 4
```

برای این سوال یک **readme** ساده در حد اینکه چطور کدتان ران شود و یک **Makefile** هم اضافه کنید.