دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

اصول سیستم‌های عامل

تمرین سری پنج

استاد درس

دکتر زهرا قربانعلی

اردیبهشت ماه ۱۴۰۴

توضیحات تمرین سری پنج:

این سوالات مربوط به فصل ۶ و ۷ می‌باشد. در صورتی که سوالی از این تمرین داشتید، می‌توانید با تدریسیاران زیر در ارتباط باشید.

- ارسلان تازیکه
- ریحانه هاشمی
- موژان بهادری
- سروین باغی

لطفا برای هرگونه سوال، به ایدی تلگرام آنها (موجود در گروه درس) مراجعه کنید.

بخش عملی این تمرین امتیازی است، در صورت تمایل می‌توانید آن رو پیاده‌سازی نمایید.

مهلت تحویل تمرین سری پنج تا تاریخ ۲ خرداد، ساعت۲۳:۵۹:۵۹ می‌باشد.

مباحث فصل ۶ و ۷:

۱- برنامه‌ای متشکل از دو نخ(*Thread*) T_1 و T_2 است. این دو نخ به طور همزمان روی دو پردازنده متفاوت در حال اجرا هستند. برنامه در حال اجرای این دو ریسه به صورت زیر است:

T_1	T_2

function thread۱()
open logFile "log.txt" in append mode
// Prepare the log entry
logEntry = "Thread A log entry at " +
getCurrentTime()
// Write the log entry to the file
write logEntry to logFile
// Close the log file
close logFile
end function
	function thread۲()
	open logFile "log.txt" in append mode
	// Prepare the log entry
	logEntry = "Thread B log entry at " +
	getCurrentTime()
	// Write the log entry to the file
	write logEntry to logFile
	// Close the log file
	close logFile
	end function

الف) این برنامه ممکن است با چه مشکلی رو به رو شود؟

ب) برای حل آن چه راهکاری پیشنهاد می‌دهید؟

۲- مانیتور mon به صورت زیر تعریف شده است.CS1 و CS2 دو بخش‌های بحرانی متفاوت و مستقل از یکدیگر می‌باشند. اگر دو فرآیند P_1 و P_2 به صورت همروند در حال اجرا باشند،کدام یکی از شروط Mutual Exclusion، Progress یا Busy Waiting نقض می‌شود؟

```
Monitor mon;
    Procedure enter1;
        Begin
            CS1;
        end;

    Procedure enter2;
        Begin
            CS2;
        end;
P1: While (True) {mon.enter1;}
P2: While (True) {mon.enter2;}
```

۳- الف) تعادل بین fairness و throughput of operations را در مسئله readers–writers تحلیل کنید.

ب) روشی برای حل این مسئله پیشنهاد دهید که باعث starvation نشود.

۴- شرودینگر یک شرکت دارد که جفت ذرات درهم‌تنیده تولید می‌کند، که سپس بسته‌بندی شده و به تولیدکنندگان کامپیوترهای کوانتومی ارسال می‌شوند. از آنجایی که این یک فرآیند پیچیده است، چندین دستگاه وجود دارد که جفت ذرات را تولید می‌کنند؛ هر دستگاه کد تولیدکننده (Producer) نشان داده شده در زیر را اجرا می‌کند. جفت ذرات تکمیل شده در بافر ذرات قرار می‌گیرند، جایی که دو مکان بافر را اشغال می‌کنند. یک دستگاه بسته‌بندی وجود دارد که یک جفت ذره را از بافر ذرات می‌گیرد و آن را برای حمل آماده می‌کند؛ دستگاه بسته‌بندی کد مصرف‌کننده (Consumer) نشان داده شده در زیر را اجرا می‌کند. برای جلوگیری از هرگونه نقض شرایط، باید قوانین زیر رعایت شود:

- یک دستگاه تولیدکننده فقط می‌تواند یک جفت ذره را در بافر قرار دهد اگر دو فضای خالی موجود باشد.
- جفت ذره باید در مکان‌های متوالی بافر ذخیره شود، یعنی یک ذره از دستگاه تولیدکننده دیگر نمی‌تواند بین ذراتی که جفت را تشکیل می‌دهند، قرار بگیرد.
- ظرفیت بافر نمی تواند بیش تر از ۱۰۰ ذره یا ۵۰ جفت ذره باشد.
- دستگاه بسته‌بندی خراب می‌شود اگر به بافر دسترسی پیدا کند و آن را خالی بیابد – فقط باید زمانی ادامه دهد که حداقل دو ذره در بافر وجود داشته باشد.

شرودینگر درباره‌ی سمافورها شنیده است اما مطمئن نیست چگونه باید از آن‌ها استفاده کند تا مطمئن باشد که قوانین بالا رعایت می‌شوند.

سمافور ها، Wait() ها و Signal() های مناسب را به کد های مصرف کننده و تولید کننده اضافه کنید تا از عملکرد صحیح وعدم بروز بن بست، اطمینان حاصل شود. حتماً مقادیر اولیه برای هر سمافوری که استفاده

می‌کنید را مشخص کنید. **به یاد داشته باشید: تولیدکنندگان متعددی وجود دارند ولی تنها یک مصرف‌کننده وجود دارد.**

سعی کنید از حداقل تعداد سمافورها استفاده کنید و محدودیت‌های اولویت غیرضروری را بیان نکنید.

```
Shared Memory
particle buffer[100];  // holds 100 particles

Semaphores and initial values: _____________________
```

```
Producer
PLoop:

   Produce pair P1, P2

   Place P1 in buffer

   Place P2 in buffer

   Go to PLoop
```

```
Consumer
CLoop:

   Fetch P1 from buffer

   Fetch P2 from buffer

   Package and ship

   Go to CLoop
```

۵- چگونگی پیاده سازی یک Mutex Lock را با استفاده از دستور compare_and_swap() در نظر بگیرید. فرض کنید که ساختار زیر برای تعریف Mutex Lock در دسترس است:

```
typedef struct {
    int available;
} lock;
```

مقدار صفر برای متغیر available نشان‌دهنده این است که قفل در دسترس است و مقدار ۱ نشان‌دهنده این است که قفل در دسترس نیست.

با استفاده از این ساختار، نحوه پیاده‌سازی توابع زیر را با استفاده از دستور compare_and_swap() نشان دهید:

```
Void acquire(lock *mutex)
Void release(lock *mutex)
```

مقداردهی های اولیه مورد نیاز را لحاظ کنید.

بخش عملی این تمرین امتیازی است، در صورت تمایل می‌توانید آن رو پیاده‌سازی نمایید.

بخش عملی:

سوال زیر رو طبق توضیحات داده شده پیاده‌سازی کنید.

۱. یک یک ریپازیتوری خصوصی (Private) در گیت‌هاب با همان نامی که در صورت سؤال ذکر شده ایجاد کنید.

۲. کدهای مربوط به سوال رو داخل اون ریپازیتوری قرار بدید.

۳. لینک ریپازیتوری رو داخل فایل پاسخ‌نامه قرار بدید و فایل پاسخ‌نامه رو آپلود کنید.

نکته مهم:

- حتما ریپازیتوری باید به صورت Private باشد. پس از اتمام ددلاین تمرین باید ریپازیتوری را Public کنید.

Synchronization Problem – TA Student Simulation

در دانشکده‌ی علوم کامپیوتر، دستیار آموزشی (TA)، در ساعات اداری به دانشجویان کارشناسی کمک می‌کند تا مفاهیم درس سیستم‌های عامل را بهتر درک کنند و به سؤالات آن‌ها پاسخ دهد. اما دفتر او کوچک است و فقط پنج صندلی برای انتظار دانشجویان دارد. بنابراین، برای مدیریت تعامل TA و دانشجویان، نیاز به پیاده‌سازی یک سیستم هم‌زمانی دقیق وجود دارد.

قوانین سیستم

۱. فقط یک دانشجو می‌تواند هم‌زمان از کمک TA بهره‌مند شود.
۲. حداکثر پنج دانشجو می‌توانند هم‌زمان در دفتر منتظر بمانند (روی صندلی‌ها).
۳. اگر صندلی خالی وجود نداشته باشد، دانشجو بازمی‌گردد و بعداً مراجعه می‌کند.
٤. اگر TA در حال استراحت باشد و دانشجویی وارد شود، او TA را بیدار می‌کند.
٥. پس از کمک به هر دانشجو، TA بررسی می‌کند که آیا دانشجوی دیگری در صف وجود دارد. اگر باشد، به او کمک می‌کند؛ در غیر این صورت، دوباره می‌خوابد.

شرح وظایف

شما باید با استفاده از مفاهیم هم‌زمانی در سیستم‌های عامل، رفتار TA و دانشجویان را شبیه‌سازی کرده و برنامه‌ای بنویسید که شامل موارد زیر باشد:

- ایجاد n نخ دانشجو که به طور تصادفی وارد دفتر می‌شوند.
- پیاده‌سازی یک نخ TA که در حالت خواب است و در صورت نیاز بیدار می‌شود.
- مدیریت صحیح صف انتظار، خوابیدن استاد، ورود و خروج دانشجویان.

جزئیات پیاده‌سازی

می‌توانید از کتابخانه‌های زیر استفاده کنید:

- pthread.h برای ساخت تردها
- semaphore.h برای استفاده از سمافور
- pthread_mutex_t برای قفل متقابل
- pthread_cond_t برای شرایط هم‌زمانی (در صورت نیاز)

شما می‌توانید این تمرین را با یکی از روش‌ها و یا ترکیبی از آن‌ها را پیاده‌سازی کنید:

۱. Mutex و Condition Variables

استفاده از pthread_mutex_t و pthread_cond_t برای کنترل صف، وضعیت TA، و دسترسی به منابع مشترک.

۲. Semaphores

مدیریت ورود و خروج دانشجویان، خوابیدن و بیدار شدن TA با sem_t، که پیاده‌سازی ساده‌تری دارد.

۳. ترکیبی از Mutex و Semaphore

ترکیب این دو روش می‌تواند انعطاف‌پذیری بیشتری در مدیریت صف و هم‌زمانی ایجاد کند.