به نام خدا

سیستم‌های عامل (پاییز ۱۴۰۱)

تمرین سوم

استاد درس:

دکتر جوادی

طراحی وجمع‌آوری سوالات:

خانم ها سروقد و میرزازاده و آقای پولاد

مهلت نهایی ارسال پاسخ:

۲۱ آذر۱۴۰۱ ساعت ۲۳:۵۹

نکته مهم: دقت کنید که تمدید نخواهیم داشت و صرفا می‌توانید ۱ تا ۵ روز از ۱۵ روز مجاز برای تاخیر ارسال تمامی تمرین‌های تئوری در این ترم را استفاده کنید. اگر بودجه ۱۵ روز شما تمام شود، به ازای هر روز تاخیر ۱۰ درصد از نمره تمرین را از دست خواهید داد.

۱ - به سوالات پاسخ دهید.

الف) با در نظر گرفتن جدول زیر و با استفاده از روش اول کوتاترین کار[1] نمودار گانت زمان‌بندی پردازنده‌ها را رسم کرده و میانگین زمان تاخیر را بدست آورید.

process	burst time
p1	4
p2	8
p3	14
p4	7

ب) فرض کنید در هر ۱۰ ثانیه یک پردازه با مدت اجرای ۵ ثانیه به صف اجرا اضافه می‌شود. همچنین پردازه هایی با مدت اجرای ۰.۵ ثانیه نیز در هر ۲ ثانیه ساخته می‌شوند. بخشی از الگوی ترتیب اضافه شدن این فرآیندها را در جدول زیر مشاهده میکنید.

Process	Arrival time	Burst time
p1	0	5
p2	1	0.5
p3	3	0.5
p4	5	0.5
p5	7	0.5
p6	9	0.5
p7	10	5
p8	11	0.5
p9	13	0.5
…	…	…

نمودار گانت اجرای فرآیندها با زمانبندی FCFS را برای مدت ۲۰ ثانیه رسم کرده و میانگین زمان انتظار فرآیندها را به دست آورید.

[1] Shortest Job First

۲ - می‌خواهیم با استفاده از دستور compare_and_swap، یک تابع برای جمع کردن دو عدد به صورت اتمیک طراحی کنیم. فرم تابع به صورت زیر است و خروجی تابع باید مقدار *p1 + p2 باشد. توجه کنید که مقدار حافظه‌ای که *p1 به آن اشاره می‌کند ممکن است هر لحظه توسط یک ترد دیگر تغییر کند و منظور از اتمیک این است که عملکرد تابع شما نباید در این صورت دچار اختلال گردد.

```
int atomic_add(int *p1, int p2);
```

برای استفاده از دستور compare_and_swap می‌توانید از تابع زیر استفاده کنید:

```
int compare_and_swap(int *value, int expected, int new_value);
```

۳- دو پردازه برای حل مسئله‌ی ناحیه‌ی بحرانی از روش زیر استفاده کرده‌اند. متغیر های S1 و S2 بین دو پردازه مشترک هستند و یک مقدار Boolean دارند که در ابتدای اجرای برنامه به صورت تصادفی مقدار دهی شده‌اند.

P2	**P1**

`while (S1 != S2);`
`//Critical Section`
`S2 = !S1`
	`while (S1 == S2);`
	`//Critical Section`
	`S1 = S2`

بررسی کنید و توضیح دهید که هر کدام از ۳ شرط Mutual Exclusion, Progress و Bounded Waiting برآورده می‌شوند یا خیر.

۴- تعداد n پردازه داریم که هر کدام از دو بخش پردازشی A و B تشکیل شده‌اند. می‌خواهیم با استفاده از سمافورها به گونه‌ای این پردازه‌ها را هماهنگ کنیم که اول قسمت A همه‌ی پردازه‌ها به طور کامل اجرا شود و بعد از آن قسمت B پردازه‌ها اجرا شود و فرض کنید تعداد پردازه‌ها را در متغیر n داریم. شبه کد زیر را به گونه‌ای تکمیل کنید که این هماهنگی ایجاد شود. سمافورهایی که استفاده کرده‌اید و مقدار اولیه آن‌ها را بالای کدتان بنویسید.

```
A()
//add your code here
B()
```

راهنمایی: می‌توانید از سمافورها و متغیرهای زیر استفاده کنید:

- count: متغیری که تعداد پردازه‌هایی که بخش A را اجرا کرده‌اند را نشان می‌دهد.
- mutex: سمافور با مقدار اولیه‌ی 1 برای تغییر دادن count.
- barrier: سمافور با مقدار اولیه‌ی 0 که تا زمانی که همه‌ی پردازه‌ها قسمت A را تمام نکرده‌اند مقدار آن بیشتر از صفر نمی‌شود.

اسم فایل ارسالی شما sid_os_hw3 باشد (sid را با شماره دانشجویی جایگزین کنید)

موفق باشید

تیم درس سیستم‌های عامل