آزمون پایان‌ترم درس اصول سیستم‌های عامل

دانشگاه صنعتی خواجه نصیرالدین طوسی- دانشکده ریاضی

تیر ماه ۱۴۰۳ – مدت زمان آزمون: ۱۲۰ دقیقه

۱- به تعداد n پروسه به طور همزمان وارد یک سیستم می‌شوند. فرض کنید الگوریتم زمان‌بندی استفاده شده از نوع Round Robin بوده و مقدار time quantum برابر با q واحد زمانی و مقدار context-switch بین دو پروسه برابر با c واحد زمانی باشند. همچنین، رابطه‌های $n \geq 2$ ، $q > 0$ و $c > 0$ نیز برقرار باشند. حال به پرسش‌های زیر پاسخ دهید:

الف) حداکثر چند واحد زمانی لازم است تا یک پروسه منتظر بماند تا نوبت به اجرای کوانتوم زمانی بعدی‌اش برسد؟

ب) به ازای $n = 11$، $q = 10$ و $c = 2$، مقدار قسمت الف چقدر خواهد شد؟

پ) فرض کنید هر یک از این n پروسه برای تکمیل شدن اجرایش به d واحد زمانی نیاز داشته باشد به طوری که $d = kq$ بوده و k عددی صحیح، مثبت و ثابت باشد. اکنون، average turnaround time را برای حالتی بیابید که در آن $c = 0$.

ت) به ازای $n = 5$، $q = 10$ و $d = 60$، مقدار قسمت پ چقدر خواهد شد؟

۲- فرض کنید در یک سیستم‌عامل فرضی الگوریتمِ زمان‌بندی استفاده شده بر مبنای اولویت (priority scheduling) است. اگر همه پروسه‌های نشان داده شده در جدول زیر همزمان وارد سیستم شده و عدد کمتر در ستون priority، نشان‌دهنده اولویت بالاتری باشد، با صرفنظر از زمان context-switch،

الف) average waiting time چقدر خواهد بود؟

ب) پس از اتمام اجرای همه پروسه‌ها، throughput چقدر خواهد بود؟

Process	CPU Burst Time (time units)	Priority
P_1	20	3
P_2	13	4
P_3	7	1
P_4	9	5
P_5	10	2

۳- فرض کنید برای پیش‌بینی next cpu burst از روش exponential averaging استفاده می‌کنیم. اکنون، به جدول زیر توجه کنید. در ردیفِ نخست این جدول، زمان‌های cpu burst که در واقعیت رخ می‌دهند نشان داده شده است. همچنین، عدد 15 در ردیف دوم نشان‌دهنده حدس اولیه در مورد next cpu burst است. با فرض $\alpha = \frac{1}{3}$،

الف) فرمول چگونگی پیش‌بینی next cpu burst را بر مبنای exponential averaging بنویسید.

ب) جدول زیر را در پاسخ‌نامه رسم کرده و با استفاده از فرمول قسمت الف آن را تکمیل نمایید..

cpu burst واقعی		6	18	5	2
next cpu burst پیش‌بینی شده	15				

۴- فرض کنید پروسه P_1 از دستور C_1، پروسه P_2 از دستور C_2 و پروسه P_3 از دستور C_3 تشکیل شده باشند و این سه پروسه به طور همروند (concurrent) در حال اجرا باشند. با استفاده از یک یا چند سمافورِ باینریِ مشترک بین پروسه‌ها و همچنین، قرار دادن دستورات signal و wait در جاهای مناسبی از این پروسه‌ها، روشی پیشنهاد دهید که تضمین کننده اجرای دستور C_1 پیش از C_2 و اجرای دستور C_2 پیش از C_3 باشد. مقدار اولیه هر یک از سمافورهای باینری را نیز معین نمایید.

۵- ساختار برنامه مربوط به فیلسوف iام در مسئله dining philosophers (DF) به صورت روبه‌رو پیاده‌سازی شده است ($0 \leq i \leq 4$). همچنین، آرایه پنج‌تایی chopstick نیز بین تمامی فیلسوف‌ها به اشتراک گذارده شده است. هر عضو این آرایه یک سمافور باینری است که وظیفه مدیریت استفاده از یکی از chopstick را بر عهده دارد. مقدارهای اولیه تمام عناصر این آرایه نیز برابر با یک است. اکنون، درستی یا نادرستی هر یک از عبارات زیر را (بدون ذکر دلیل) بیان نمایید.

```
while (true) {

        wait(chopstick[i]);
        wait(chopstick[(i+1) % 5]);
        . . .
        /* eat for a while */
        . . .
        signal(chopstick[i]);
        signal(chopstick[(i+1) % 5]);
        . . .
        /* think for awhile */
        . . .

}
```

الف) این شیوه از پیاده‌سازی مسئله DF خاصیت mutual exclusion را برآورده می‌سازد.

ب) در این شیوه از پیاده‌سازی مسئله DF ممکن است برخی از فیلسوف‌ها دچار starvation شوند.

پ) در این شیوه از پیاده‌سازی مسئله DF ممکن است زیرمجموعه‌ای از فیلسوف‌ها دچار deadlock شوند.

۶- فرض کنید که دستور اتمیک compare_and_swap به شکل زیر تعریف شده باشد.

```
int compare_and_swap(int *value, int expected, int new value) {

        int temp = *value;

        if (*value == expected)
                *value = new value;

        return temp;
}
```

```
while (true) {

        while (compare and swap(&lock, 0, 1) != 0);
        ...
        /* critical section */
        ...
        lock = 0;
        ...
        /* remainder section */
        ...
}
```

حال دو پروسه را در نظر بگیرید که به صورت همروند اجرا می‌شوند و ساختار هر یک از پروسه‌ها به شکل روبه‌رو است. در اینجا، lock متغیری از نوع global با مقدار اولیه صفر است.

توضیح دهید که چگونه یکی از دو پروسه در رقابت با دیگری می‌تواند به critical section وارد شود و دیگری از ورود به آن باز بماند.

۷- برنامه روبه‌رو را یک بار اجرا می‌کنیم و اجرای آن به اتمام می‌رسد. با فرض اینکه هر سه فراخوانی دستور ()fork بدون اِشکال اجرا شود، به پرسش‌های زیر (بدون ذکر دلیل) پاسخ دهید.

```
#include <unistd.h>
#include <stdio.h>

int a = 0;

void main() {
        fork();
        a = a + 1;
        fork();
        a = a + 1;
        if (fork() == 0) {
                printf("Hello\n");
        } else {
                printf("Goodbye\n");
        }
        a = a + 1;
        printf("a is %d\n", a);
}
```

الف) کلمه Hello چند بار در خروجی چاپ می‌گردد؟

ب) کلمه Goodbye چند بار در خروجی چاپ می‌گردد؟

پ) چند بار مقدار a در خروجی چاپ خواهد شد؟

ت) در نتیجه پروسه‌های ایجاد شده، آیا مقدارهای متفاوتی از a توسط این پروسه‌ها چاپ می‌شود یا همه مقدارهای a چاپ شده یکسان هستند؟

۸- بدون ذکر دلیل، صرفاً، درستی یا نادرستی هر یک از موردهای زیر را معین نمایید.

الف) اغلب، اجرای یک پروسه شامل تعداد زیادی CPU burst با مدت زمان کوتاه و تعداد کمی CPU burst با مدت زمان طولانی‌تر است.

ب) روش زمان‌بندی preemptive تضمین می‌کند که اجرای پروسه‌هایی که به داده‌های مشترک دسترسی دارند هیچ‌گاه، منجر به race condition نمی‌شود.

پ) اگر برای اجرای تعدادی پروسه از یک الگوریتم زمانبندی استفاده کنیم و average turnaround time و average waiting time را به ترتیب برابر با t_t و t_w نشان دهیم، ممکن است حالتی پیش آید که در آن، رابطه $t_t < t_w$ برقرار باشد.

ت) فرض کنید هشتاد درصد برنامه‌ای را بتوان به صورت موازی اجرا نمود. بر اساس قانون Amdahl، اگر تعداد Coreهای تخصیص داده شده برای اجرای این برنامه از ۱ به ۶ افزایش یابد، میزان افزایش سرعت اجرای برنامه (Speedup) حداکثر چهار و نیم برابر خواهد شد.

ث) در حالت کلی، الگوریتم زمان‌بندی FCFS برای سیستم‌های interactive مناسب نیست.

ج) سیستمی را در نظر بگیرید که می‌تواند در دو حالت پایدار (steady state) S و S' قرار داشته باشد. در حالت S، به تعداد λ پروسه جدید در هر ثانیه وارد سیستم شده و به طور متوسط به تعداد n پروسه در هر لحظه در صف انتظار (برای اجرا شدن) وجود دارد. در حالت S'، به تعداد λ' پروسه جدید در هر ثانیه وارد سیستم شده و به طور متوسط n' پروسه در هر لحظه در صف انتظار وجود دارد. اگر $n < n'$ و $\lambda > \lambda'$ باشند، در اینصورت، حتماً، به طور متوسط، زمان انتظار هر پروسه (در صف انتظار) در S نسبت به S' کمتر خواهد بود.

چ) پروسه P را در نظر بگیرید که به تعدادی فایل دسترسی دارد. فرض کنید P از دو thread با نام‌های T و T' تشکیل شده باشد. دراینصورت، T و T'، هر دو به فایل‌های P دسترسی دارند. اما، T پشته (stack) مخصوص به خود و T' نیز پشته مخصوص به خود را دارند.

ح) مدل two-level که مدل ویژه‌ای از تناظر بین user threads و kernel threads را بیان می‌کند، تلفیقی از مدل‌های one-to-one و many-to-many است.

خ) هر upcall ایجاد شده توسط یک kernel thread را یک upcall handler پاسخ می‌دهد. هر upcall handler برای اجرا شدن، نیاز به یک LWP (Lightweight Process) دارد.