دانشکده‌ی مهندسی کامپیوتر

سیستم‌های عامل
تمرین‌های سری دوم

علی حیدری

۱۹ اردیبهشت ۱۳۹۸

فهرست مطالب

۱ پرسش اول

هنگامی که یک پردازه یک پردازه‌ی دیگر را به وسیله‌ی دستور `fork()` ایجاد می‌کند کدام یک از موارد زیر بین پردازه‌ی پدر (parent) و فرزند (child) به اشتراک گذاشته می‌شود و برای موارد دیگر چه اتفاقی رخ می‌دهد؟ (share می‌شود)

- Stack
- Heap
- Shared Memory Segments

پاسخ. Shared Memory Segments
دلیل: زمانی که `fork()` اتفاق می‌افتد یک کپی از همه‌ی صفحه‌ها متناظر با فرایند پدر ایجاد می‌شود و در یک مکان حافظه‌ی مجزا توسط سیستم‌عامل در فرایند فرزند بارگذاری می‌شود. [۱]

۲ پرسش دوم

آ) الگوریتم‌های Scheduling زیر را توضیح دهید.

- Non Pre-Emptive, First Come, First Serve

پاسخ. به طور کلی در الگوریتم‌هایی که Non Pre-Emptive هستند زمانی که فرایندی وارد حالت «در حال اجرا» می‌شود آن فرایند حذف نمی‌شود مگر آن که زمان سرویس آن پایان یابد. [۲]
در الگوریتم FIFO هر فرایندی که زودتر وارد شود زودتر سرویس داده می‌شود. [۳]

- Round Robin

پاسخ. این الگوریتم یک الگوریتم غیر انحصاری زمان‌بندی است. هر فرایند یک زمان ثابت برای اجرا شدن دارد که به آن **quantum** می‌گویند. زمانی که یک فرایند در مقطع زمانی تعیین شده اجرا می‌شود قبضه می‌شود تا فرایند دیگری اجرا شود و آن فرایند هم به همان مقدار زمان برای اجرا خواهد داشت و این روند ادامه خواهد داشت. به بیان ساده‌تر اگر مدت زمان ثابتی را در نظر بگیریم در هر زمان فقط یک فرایند می‌تواند وجود داشته باشد و توزیع مقطع زمانی میان فرایندها می‌تواند براساس الگوریتم FIFO یا FIFO اولویت‌دار باشد. [۴]

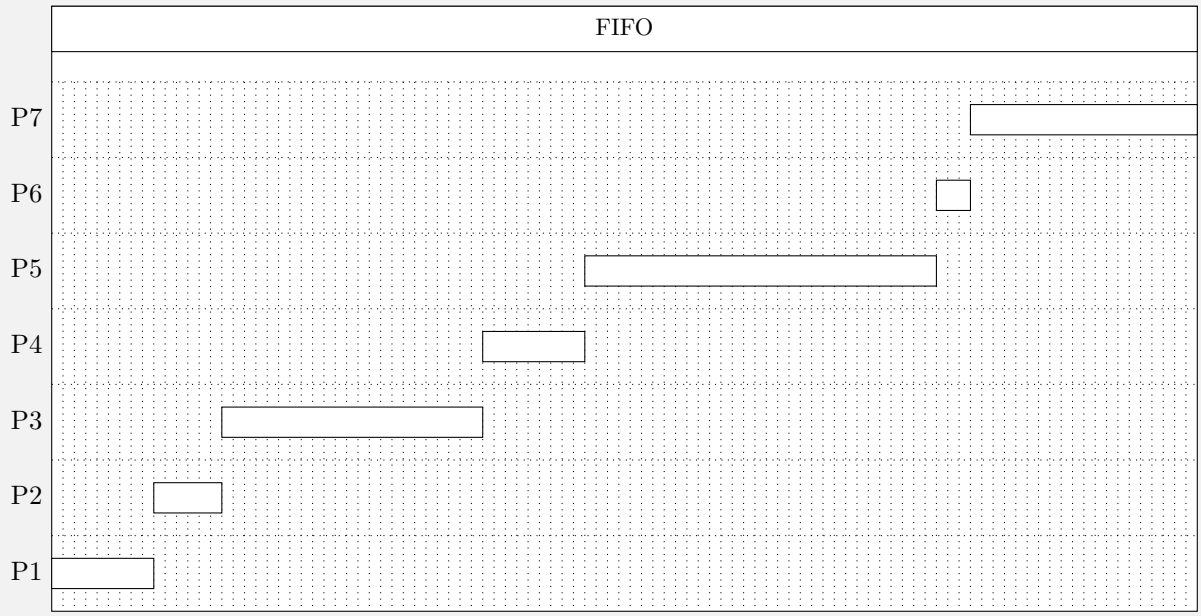
- Shortest Job First

پاسخ. این الگوریتم یک الگوریتم غیر انحصاری است که در آن هر فرایندی که زمان سرویس کم‌تری داشته باشد زودتر سرویس داده می‌شود. این الگوریتم کم‌ترین زمان انتظار را داراست و در موقعیت‌هایی کاربردی است که زمان سرویس فرایندها را می‌دانیم. [۴]

ب) با توجه به جدول زیر و پردازه‌ها و Burst Time های داده شده مقدار میانگین زمان مورد نیاز هنگام استفاده از هر کدام از الگوریتم‌های بالا را به دست آورید. (مقدار quantum را ۸ در نظر بگیرید)

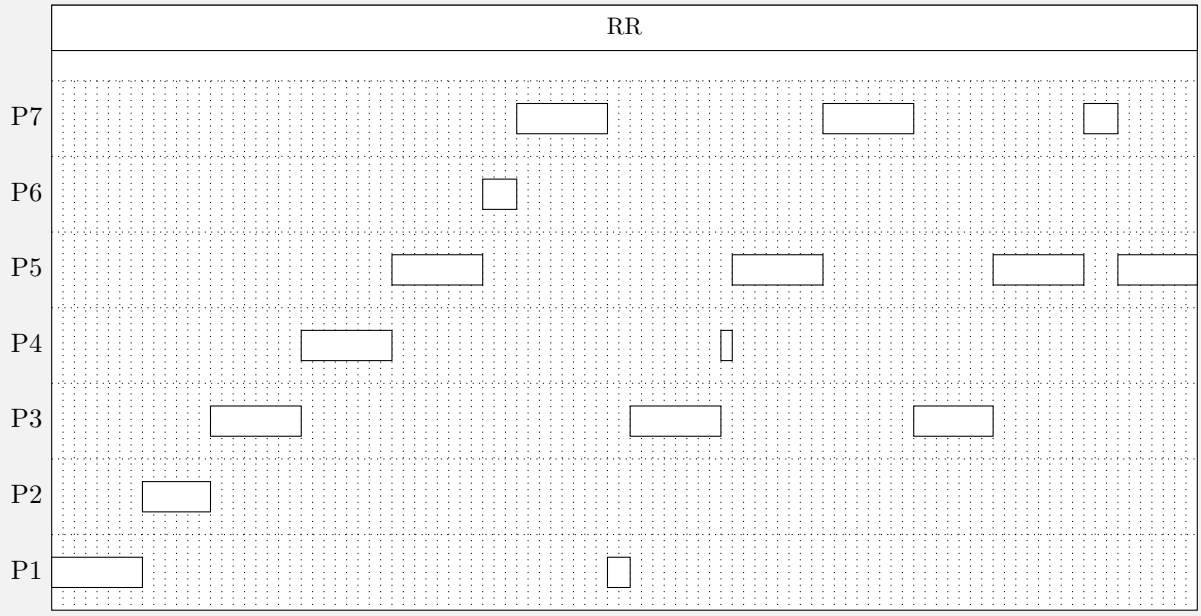
Process	Burst Time
P1	10
P2	6
P3	23
P4	9
P5	31
P6	3
P7	19

پاسخ. با فرض ورود همه‌ی فرایندها در زمان صفر داریم:
نمودار Gantt با الگوریتم FIFO برای فرایندهای بالا به صورت زیر است:



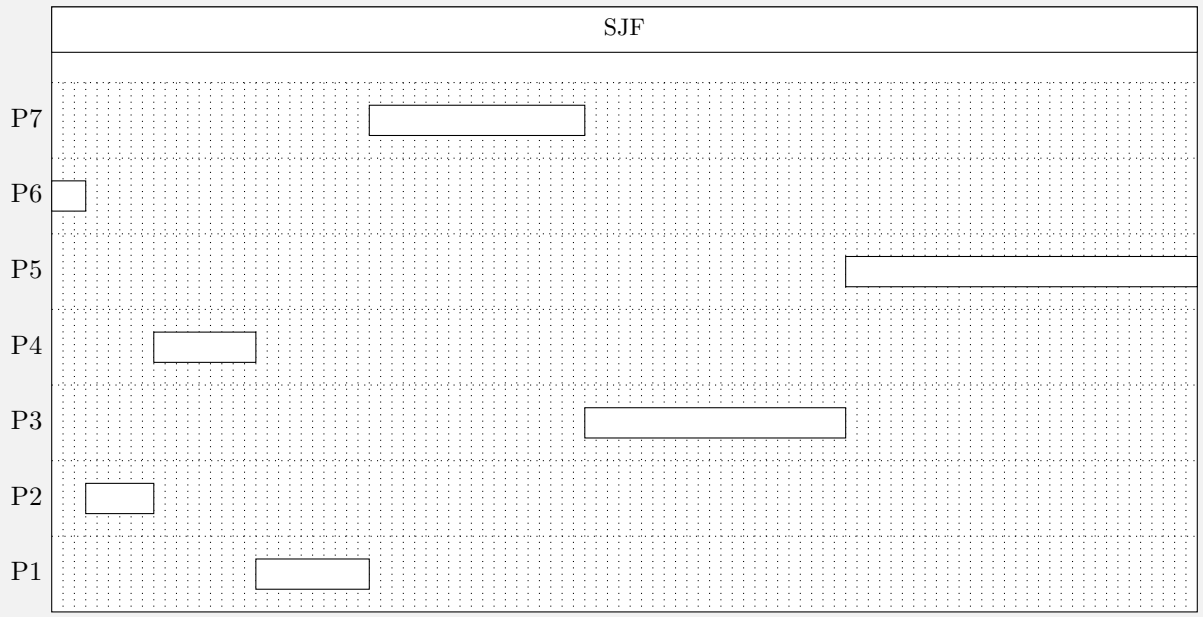
$$\text{Avrage waiting time} = \frac{\sum_{i=۰}^{n} W_{p_i}}{n} = \frac{۱۰ + ۱۶ + ۳۹ + ۴۸ + ۷۹ + ۸۲ + ۱۰۱}{۷} = ۳۹/۱۴$$

نمودار Gantt با الگوریتم RR برای فرایندهای بالا به صورت زیر است:



$$\text{Avrage waiting time} = \frac{\sum_{i=۰}^{n} W_{p_i}}{n} = \frac{۴۱ + ۸ + ۶۰ + ۵۱ + ۷۰ + ۳۸ + ۷۵}{۷} = ۴۹/۰۰$$

نمودار Gantt با الگوریتم SJF برای فرایندهای بالا به صورت زیر است:



$$\text{Avrage waiting time} = \frac{\sum_{i=0}^{n} W_{p_i}}{n} = \frac{0+3+9+18+28+47+70}{7} = 25/00$$

ج) با توجه به نتایج به دست آمده تحلیل کنید که به عنوان یک طراح سیستم‌عامل از کدام یک از الگوریتم‌های بالا استفاده می‌کنید.

پاسخ. در صورتی که بخواهیم کم‌ترین زمان انتظار را داشته باشیم از الگوریتم SJF استفاده می‌کنیم. [۴]

در صورتی که بخواهیم کم‌ترین زمان انتظار را داشته باشیم از الگوریتم SJF استفاده می‌کنیم. [۴]

الگوریتم RR برای Time sharing مناسب است اما برای سیستم‌عامل‌های بلادرنگی [۱] که محدودیت روی ددلاین دارد مناسب نیست. [۵]

نتیجه‌گیری: در صورتی که بتوان مشکل گرسنگی را برای الگوریتم SJF مدیریت و حل کرد این الگوریتم الگوریتم مناسبی است در غیر این صورت الگوریتم RR مناسب‌تر به نظر می‌رسد. البته انتخاب الگوریتم بسیار وابسته به نوع سیستم مورد بررسی است.

۳ پرسش سوم

فرض کنید که ۲ نخ [۲] به صورت همزمان [۳] قطعه کد C زیر را اجرا می‌کنند و به متغیرهای مشترک‌های [۴] a و b و c دسترسی دارند.

Listing 1: Initialization

```
int a = 4;
int b = 0;
int c = 0;
```

[2] Thread
[3] Concurrently
[4] Shared Variables

Listing 2: Thread 1

```
if(a < 0){
    c = b - a;
} else {
        c = b + a;
    }
```

Listing 3: Thread 2

```
b = 10;
c = -3;
```

مقدار نهایی متغیر c پس از اجرای کامل هر دو نخ [۵] می‌تواند چه عددی باشد؟ فرض کنید که عملیات‌های read/write همگی atomic بوده و ترتیب اجرای دستورات توسط کامپایلر به همان ترتیبی است که در کد نوشته شده است.

پاسخ. ترتیب اجرای دستورات به صورتی که در کد هر نخ نمایش داده شده است می‌باشد. بنابراین بسته به این که کدام دستور زودتر اجرا شود نتیجه متفاوت خواهد بود.

Listing 4: Thread 1

```
if(a < 0){      // first command
    c = b - a;  // second command
} else {
        c = b + a;  // second command
    }
```

Listing 5: Thread 2

```
b = 10; // first command
c = -3; // second command
```

بررسی دو حالت ممکن:

Listing 6: Case 1

```
1 if(a < 0)
2 b = 10;
3 c = b + a;
4 c = -3; // final value of c will be -3
```

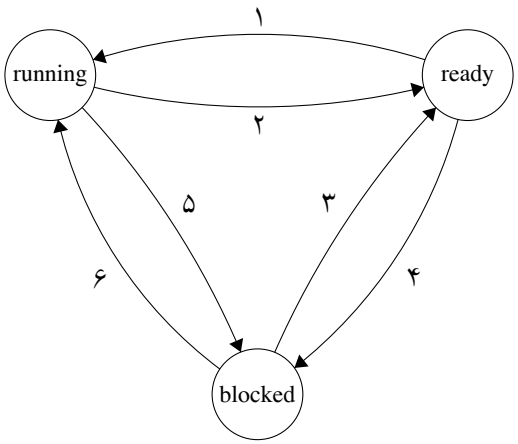
Listing 7: Case 2

```
1 b = 10;
2 if(a < 0)
3 c = -3;
4 c = b + a; // final value of c will be 14
```

بنابراین مقدار نهایی متغیر c می‌تواند ۳- یا ۱۴ باشد.

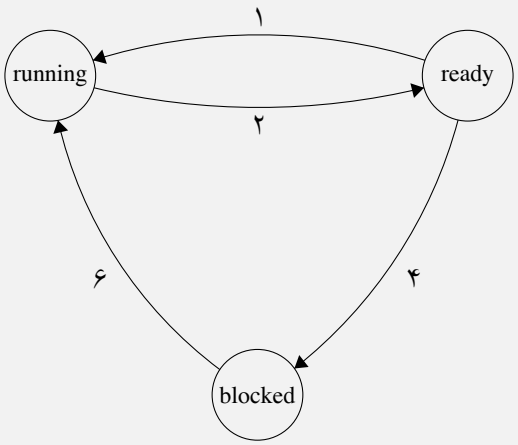
۴ پرسش چهارم

به صورت خلاصه بیان کنید که چه اتفاقی باعث جابجایی یک نخ [۶] بین هر کدام از ۳ وضعیت زیر می‌شود. همچنین علت هر کدام از فلش‌ها را بیان کنید و در صورت عدم رخداد هر کدام جای آن را خالی بگذارید.



[5] Thread
[6] thread

پاسخ.
۱: وقفه
۲: توزیع توسط زمان‌بند
۴: فراخوانی ورودی/ خروجی یا رخداد انتظار
۶: فراخوانی ورودی/خروجی یا رخداد تکمیل



مراجع

[۱] Operating Systems General Concepts

[۲] What is pre-emptive and non-preemptive scheduling?

[۳] FIFO (computing and electronics)

[۴] Operating System Scheduling algorithms

[۵] Advantages of Round Robin Scheduling?