دانشکده ی مهندسی کامپیوتر

دانشگاه صنعتی امیرکبیر

تمرین ششم درس سیستم عامل

مهلت تحویل ساعت ۲۳:۵۹ روز ۱۶ آبان ۹۹

تمرینات را انفرادی حل کرده و در سایت مودل (courses.aut.ac.ir) با قالب زیر بارگزاری نمایید:

StudentID_Name_Last Name

درصورت داشتن سوال درمورد این تمرین، سوال خود را با موضوع تمرین ۶ با ایمیل زیر درمیان بگذارید:

osfall2020@gmail. com

توجه : نمره این تمرین امتیازی است و فرصتی برای جبران تمارین گذشته و بعدی می باشد. با این وجود حل سوالات برای آشنایی بهتر با این مباحث و نمونه سوالات آن پیشنهاد اکید ما است.

۱- با استفاده از قانون آمدال میزان تسریع یک سیستم که ۵۰ درصد آن به صورت موازی کار می کند را در دو حالت زیر به‌دست آورید.

الف) از ۲ هسته پردازشی استفاده می کند

ب) از ۴ هسته پردازشی استفاده می کند

۲- پس از اجرای قطعه کد زیر ، بیشترین و کمترین مقداری که در خروجی دیده می شود را همراه با ذکر دلیل بیان کنید.

```
interleave () {
        pthread_t th0, th1, th2;
        int count=0;
        pthread_create(&th0 , 0 , test , 0);
        pthread_create(&th1 , 0 , test , 0);
        pthread_create(&th2 , 0 , test , 0);
        pthread_join(th0 , 0);
        pthread_join(th1 , 0);
        pthread_join(th2 , 0);
        printf(count);
}
test () {
for (int j=0; j < 1000 ; j++)
        count=count+1;
        }
```

۳- در ادامه یک قطعه کد برای تعمیم الگوریتم ***Peterson*** برای سه فرایند ارائه شده است. این راه حل را با توجه به سه شرط انحصار متقابل ، پیشرفت و انتظار محدود بررسی کنید.

```
void enter_region(int process) {
        int other1 = (process + 1) % 3; /* other1 and other 2 are the ids */
        int other2 = (process + 2) % 3; /* of the other two processes     */
        interested[process] = TRUE;
        turn = other1;
        while (turn != process && (interested[other1] || interested[other2]))
                ; /* do nothing */
        }
void leave_region(int process) {   interested[process] = false; }
```

۴- برای مسئله انحصار متقابل راه حل نرم‌افزاری زیر پیشنهاد شده است.

الف) صحت شرط مذکور را در این راه حل بررسی کنید و نحوه کلی عملکرد الگوریتم را توضیح دهید.

ب) صحت شروط پیشرفت و انتظار محدود را هم بررسی کنید

```
wants_to_enter : array of 2 booleans
turn : integer

wants_to_enter[0] ← false
wants_to_enter[1] ← false
turn ← 0   // or 1
```

```
p0:
wants_to_enter[0] ← true
  while wants_to_enter[1] {
    if turn ≠ 0 {
      wants_to_enter[0] ← false
      while turn ≠ 0 {
        // busy wait
      }
      wants_to_enter[0] ← true
    }
  }

  // critical section
  ...
  turn ← 1
  wants_to_enter[0] ← false

  // remainder section
```

```
p1:
  wants_to_enter[1] ← true
  while wants_to_enter[0] {
    if turn ≠ 1 {
      wants_to_enter[1] ← false
      while turn ≠ 1 {
        // busy wait
      }
      wants_to_enter[1] ← true
    }
  }
   // critical section
  ...
  turn ← 0
  wants_to_enter[1] ← false

  // remainder section
```