Department of
Computer Engineering

Homework 3

Fall 2023

Dr. Javadi

Deadline: 1402/9/19

۱- فرض کنید پردازه های زیر را داریم:

	Arrival time	Priority number	CPU burst
P0	5	1	20
P1	0	4	40
P2	0	2	20
P3	13	3	5
P4	30	4	5
P5	17	1	10

الف)برای زمانبندهای FCFS، SJF، SRT و اولویت پسگیر نحوه تخصیص CPU به پردازه ها را نشان دهید.(اگر در زمانبند اولویت پسگیر دو پردازه شرایط یکسان برای انتخاب شدن داشتند آنی را انتخاب کنید که زودتر آمده باشد،همچنین در سایر زمانبند ها آنی را انتخاب کنید که پر اولویت تر است)

ب)میانگین طول عمر،زمان انتظار،زمان پاسخ و بازده CPU پردازه ها را،برای هر یک از زمانبند های فوق محاسبه کنید.

۲- با در نظر گرفتن زمان context switch=0 در هر مورد، نحوه زمانبندی پردازه ها را با یک نمودار Gantt نشان دهید:

الف) یک زمانبند چند لایه ای داریم که در آن پردازه ها دوبار وقت دارند در لایه اول زمانبندی شوند،اگر کار آنها به پایان نرسید وارد لایه دوم شده و یک بار هم وقت دارند آنجا زمانبندی شوند و در نهایت اگر کار آنها همچنان به پایان نرسید وارد لایه آخر شده و تا زمانی که کار پردازشی آنها به اتمام برسد در آنجا زمانبندی می شوند،داریم:

لایه اول: RR با کوانتوم زمانی 4

لایه دوم: RR با کوانتوم زمانی 8

لایه سوم: FCFS

و پردازه ها مطابق جدول روبرو باشند:

	Arrival time	CPU burst
P0	0	13
P1	0	10
P2	8	17
P3	24	40
P4	40	16
P5	80	4

و همه جا اولویت تخصیص پردازنده با پردازه ای باشد که کار در لایه بالاتر دارد و زمانبند غیرپسگیر(non preemptive) باشد.

ب) یک زمانبند کاملا پسگیر(preemptive) داریم که بر اساس اولویت کار می کند و اگر دو پردازه اولویت یکسان داشته باشند بر اساس SRT عمل می کند. در این زمانبند، به ازای هر 20 واحد زمانی،در صورتی که کار پردازه به اتمام نرسیده باشد، یک واحد از عدد اولویت پردازه ها کم شود و پردازه ها مطابق جدول زیر هستند:

	Arrival time	CPU burst	Priority number
P0	45	9	2
P1	0	15	4
P2	10	30	3
P3	15	20	3
P4	50	26	4

۳- شروط Mutual exclusion، Progress و Bounded Waiting را براي الگوريتم‌هاي زير بررسی کرده و با دلیل توضیح دهید. فرض کنید i و j اعدادی هستند که اندیس پردازه را مشخص می کنند.

```
do {
   flag[i]=true;
   turn=j;
   while(!flag[j]||turn==j);
       // critical section
   flag[i]=flase;
       // remainder section
} while(true);
```

i

```
do {
   flag[j]=true;
   turn=j;
   while(!flag[i]&&turn==j);
       // critical section
   flag[j]=flase;
       // remainder section
} while(true);
```

ii

٤- دو پردازه براي حل مسئله ی ناحیه بحرانی از روش زیر استفاده کردند. متغیر های s1 و s2 بین دو پردازه مشترک هستند و یک مقدار Booleanدارند که در ابتدای اجرای برنامه به صورت تصادفی مقدار دهی شده اند.

```
While (S1 != S2);
//CRITICAL SECTION
S2 = !S1;
```

```
While( S1 == S2);
//CRITICAL SECTION
S2 = S1;
```

الف) بررسی کنید و توضیح دهید که هر کدام از سه شرط Progress، Mutual exclusion و Bounded waiting برآورده می شود یا خیر.

ب) راه حلی برای عدم نقض هرکدام از شرط های بالا ارائه دهید.

۵- الف) بدون استفاده از قفل و تنها با استفاده از دستور compare_and_swap تابع زیر را به گونه ای کامل کنید که به صورت اتمی عملیات تفریق را انجام دهد. منظور از عملیات تفریق کم شدن مقدار v از حافظه ای که p به آن اشاره دارد. سپس توضیح دهید تضمینی برای انجام شدن این عملیات وجود دارد یا خیر.

```
Int sub(int *p, int v){
    //TODO
    Return *p – v;
}
```

برای استفاده از دستورcompare_and_swap از تابع زیر استفاده کنید:

```
bool compare_and_swap(int *p, int old, int new){
    If(*p != old){
        Return false;
    }
    Else{
        Return true;
    }
}
```

ب) حال توضیح دهید compare_and_swap چگونه کنترل همزمانی را در برنامه‌های چند نخی را بهبود می‌بخشد، و در چه مواقعی کاربرد دارد؟

٦- در این تمرین عملی میخواهیم نحوه عملکرد scheduler های لینوکس را مورد بررسی قرار دهیم. به صورت کلی در لینوکس سه نوع scheduler وجود دارد:

1. SCHED_OTHER

2. SCHED_FIFO

3. SCHED_RR

در زبان c میتوانیم مشخص کنیم که با چه scheduler ای برنامه‌مان اجرا شود. برای این تمرین نیاز داریم که دو پردازه داشته باشیم و آنها را با scheduler های مختلف تست کنیم.

برای نوشتن این برنامه به کتابخانه های زیر نیاز داریم :

```c
#include <sched.h>
#include <stdio.h>
#include <sys/resource.h>
#include <unistd.h>
```

برای تغییر scheduler policy برنامه، از دستور زیر استفاده می‌شود:

```c
struct sched_param param;
sched_setscheduler(getpid(), SCHED_FIFO, &param)
```

sched_param پارامترهای scheduler را مشخص می‌کند. برای تغییر اولویت پردازه، از این استراکت استفاده می‌کنیم:

```c
param.sched_priority = PRIORITY_NUM;
```

نکته : در اینجا هر چه مقدار بیشتر باشد، اولویت بالاتر است. و مقداری که اینجا تعیین میشود با مقداری که در ستون priority پردازه در دستور top میبینید متفاوت است.

با استفاده از این دستورات، برنامه ای بنویسید و سناریو های زیر را برای دو پردازه همزمان تست کنید (برنامه ای بنویسید که زمان زیادی داشته باشد که بتوانید خروجی ها را مقایسه کنید. برای نمونه می توانید از قطعه کد زیر استفاده کنید:)

```c
int n = 0;
while(1) {
  n++;
  if (!(n % 10000000)) {
    printf("FIFO running (n=%d)\n" ,n);
  }
}
```

نکته: تغییر scheduler یک پردازه نیازمند دسترسی root است بنابراین برنامه را با sudo اجرا کنید.
نکته : دو پردازه را باید را باید روی یک cpu اجرا کنیم تا نتیجه مدنظر را ببینیم. برای اجرای یک پردازه روی یک cpu core، از دستور زیر استفاده میکنیم :

```
sudo taskset -c 0 ./FIFO.o
```

سناریو ۱: پردازه اول : SCHED_FIFO و priority=1
پردازه دوم : SCHED_FIFO و priority=1

سناریو ۲: پردازه اول : SCHED_RR و priority=1
پردازه دوم : SCHED_RR و priority=1

سناریو ۳: پردازه اول : SCHED_FIFO و priority=1
پردازه دوم : SCHED_FIFO و priority=2

(امتیازی)

نکات مهم:

*مهلت ارسال تمرین ساعت 59: 23 روز 1402/9/19 می باشد،با توجه به مهلت تجمیعی هفت روز تاخیر مجاز برای تمارین،امکان تمدید تمرین وجود ندارد،بنابراین توصیه می شود ارسال پاسخ های خود را به ساعات پایانی موکول نکنید.

*در صورت کشف هر گونه تقلب بار اول برای هر دو طرف نمره صفر لحاظ می شود و از دفعات بعد مشمول جریمه نیز می گردند.

*پاسخ های خود را در قالب یک فایل pdf یا zip با فرمت OS_HW3_StudentNumber.pdf یا OS_HW3_StudentNumber.zip ارسال نمایید،مانند:

OS_HW3_9931005.pdf

*در تمام سوالات پیاده سازی توابع fork و ... را مطابق با مطالب آموزش داده شده در کلاس در نظر بگیرید و اجرای آنها را موفقیت آمیز و بدون خطا لحاظ کنید.

*هر گونه سوال خود را می توانید در گروه پرسش و پاسخ درس از ما بپرسید.

Useful links for study:

https://en.wikipedia.org/wiki/Scheduling_(computing)
https://man7.org/linux/man-pages/man2/sched_setscheduler.2.html
https://www.geeksforgeeks.org/introduction-of-process-synchronization/