به نام خدا

پاسخنامه تکلیف سوم درس سیستم عامل- همگام‌سازی

دانشگاه صنعتی اصفهان- ترم اول ۱۴۰۰

۱- انحصار متقابل: برقرار نیست اگر ابتدا P1 شروع کند، می‌تواند از حلقه blocked[1-id] در آید (زیرا این مقدار برای پروسس صفرم هنوز false است) حال قبل از اینکه turnرا برابر ۱ کند، P0 اجرا شود. اکنون p0 اصلاً وارد حلقه اول نمیشود زیرا همچنان turn=0 است پس وارد ناحیه بحرانی می‌شود. در این حالت اگر نوبت اجرا دوباره به پروسس P1 داده شود trun را به یک تغییر می‌دهد و وارد ناحیه بحرانی می‌شود. بنا بر این هر دو پروسس الان در ناحیه بحرانی هستند.

پیشرفت: در ابتدا، turn=0 است لذا پروسس صفرم وارد CS می‌شود پروسس یکم در حلقه (while(blocked[0] می‌ماند(blocked[0]=true). پس از اینکه پروسس صفرم از CS خارج شد، blocked[0] را false می‌کند پس پروسس یکم از حلقه مذکور خارج می‌شود و turn را یک می‌کند که باعث می‌شود حلقه while(turn!=id)هم برای او شکسته شود و وارد CS شود. در این حالت پروسس صفرم اگر بخواهد دوباره وارد CS شود در صورتی که پروسس یکم از CS خارج شده باشد و blocked[1] را false کرده باشد می‌تواند وارد CS شود پس به همین صورت امکان ورود هردو در صورت تمایل یکی پس از دیگری وجود دارد و پیشرفت برقرار است.

انتظار محدود:

شرط بن بست: بن‌بست وجود ندارد چون امکان ندارد شرایطی پیش آید که هر دو پروسس پشت یکی از حلقه ها بمانند و هر دو نتوانند وارد CS شوند زیرا turn یا یک است و یا صفر که باعث می‌شود بالاخره یکی از دو پروسس شرط while اول را رد کنند

شرط گرسنگی: اگر پروسس صفرم وارد CS شود و سپس از آن خارج شود، در صورتی که پروسس یکم تمایلی به ورود به CS نداشته باشد، blocked[1] برابر false مانده و پروسس صفرم دوباره فرصت آن را پیدا میکند که وارد CS شود حال اگر پروسس یکم تمایل به ورود داشته باشد، در حلقه blocked[0] میماند، در این حالت اگر نوبت اجرا دوباره به پروسس صفرم داده شود (و پروسس اول در حال اجرا نباشد) پس از خروج از CS با اینکه blockedخود را false می‌کند اما چون پروسس اول در حال اجرا نیست پشت همان whileهمچنان مانده، در این حال اگر پروسس صفرم به اجرا ادامه دهد برای بار بعدی می‌تواند وارد CS شود چون همچنان نوبت دست خودش است. این کار میتواند نامحدود ادامه پیدا کند و پروسس یکم گرسنه بماند.

۲-

```
//code for guest
lock(m)
guest_count++;
if(guest_count==N)
   signal(cv_host)
wait(cv_guest,m)
signal(cv_guest)
unlock(m)
enterHouse()
```

readLock:

```
lock(mutex)
while(writer_present || writers_waiting > 0)
        wait(reader_can_enter,mutex);
readcount++
unlock(mutex)
```

readUnlock:

```
lock(mutex)
readcount--
if(readcount==0)
        signal(writer_can_enter)
unlock(mutex)
```

writeLock:

```
lock(mutex)
writer_waiting++
while(readcount > 0 || writer_present)
        wait(writer_can_enter, mutex)
writer_waiting--
writer_present = true
unlock(mutex)
```

writeUnlock:

```
lock(mutex)
writer_present = false
if(writer_waiting==0)
        signal_broadcast(reader_can_enter)
else
        signal(writer_can_enter)
unlock(mutex)
```

۴- به صورت مشابه و متقارن تابع north نوشته می‌شود

```
Sem mutex =1, south_permit = 0 , north_permit = 0;

int north_passing=0, north_passed = 0, north_waiting = 0;

int south_passing=0, south_passed = 0, south_waiting = 0;

void south( ){

        wait(mutex)
        if (north_passing==0 && north_waiting==0){
                south_passing ++;
                signal( south_permit );
        } else if (north_waiting >0 && north_passing ==0 && south_passing+south_passed < 5 ){
                south_passing ++;
                signal(south_permit);
        }
        else
                south_waiting ++;
        signal (mutex)

        wait(south_permit)

        //pass the bridge

        wait(mutex)
        south_passing - - ;
        south_passed ++ ;
        if ( south_passing==0 && south_waiting==0 && north_waiting >0){
                south_passed = 0;
                n = min(5, north_waiting)
                for( i=0; i<n;i++){
                        signal(north_permit)
                        north_passing + +;
                        north_waiting - -;
                }
        }else if (north_waiting==0 && north_passing == 0 && south_waiting>0){
                signal(south_permit)
                south_waiting - -;
                south_passing ++;

        }
        signal(mutex);
}
```