**12 Nisan 2021**

|  |  |
| --- | --- |
| **No**  **504201574** | **Ad Soyad Gul Eda Aydemir** |

**PARALEL PROGRAMLAMA**

**ARASINAV**

**9:00-10:30**

**Soru 1:** **(20 puan)**

P0 ve P1 prosesleri, karşılıklı dışlama koşullarını gerçeklemek üzere, aşağıda yer alan kodu verilen **ilk değerler** ile yürütmektedirler. Yerel değişkenler olan **“i” ve “j”**, kodu yürütmekte olan prosesin indisine uygun olan değerlere sahiptirler: P0 için i=0, j=1 ve P1 için i=1, j=0.

boolean bekliyor[2] = { false, false };

int sıra = 0;

**mx\_begin: mx\_end:**

1. bekliyor [i] = true; 1. bekliyor [i] = false;

2. while (sıra != i) {

3. while (bekliyor [j]);

4. sıra = i; }

Verilen çözümü inceleyerek, karşılıklı dışlama koşullarını yerine getirip getirmediğini belirtin. Yanıtınızın nedenini bir örnek senaryo ile gösterin.

boolean bekliyor[2] = { false, false };

int sıra = 0;

pi

--

1. bekliyor [0] = true; 1. bekliyor [0] = false;

2. while (sıra != 0) {

3. while (bekliyor [1]);

4. sıra = 0; }

pj

--

1. bekliyor [1] = true; 1. bekliyor [1] = false;

2. while (sıra != 1) {

3. while (bekliyor [0]);

4. sıra = 1; }

Senaryo:

pi geldi, bekliyor 0 true yapti, sira 0 oldugu icin dislama yapmadan kritik bolgeye giris yapti, pj ise pi kb deyken geldi diyelim, bekliyor 1 true oldu, sira 1 olmadigi icin while girdi, ikinci while ile bekliyor 0 true oldugu icin orada busy waiting yapacaktir. bu sirada p0 mxend’e girdiginde bekliyor 0 i false yapacak dolayisiyla pj processimiz whiledan donup sirayi kendine cekip kb’ye girecektir. Kisaca, bekliyor degerini ilk set eden, kb’yi aliyor.Ikisi de true olursa sira hangisindeyse o devam edecektir.

Sonuc:

Dolayisiyla, kb’de tek bir process bulunabildiginden evet, karsilikli dislama kosulunu yerine getirecektir.

**Soru 2:** **(20 puan)**

Bir sistemde, A ve B tipinden olmak üzere iki tip proses bulunmaktadır. A tipinden tüm prosesler aynı kodu yürütmekte, B tipinden olan tüm prosesler de aynı kodu yürütmekteler. Prosesler aşağıda yer alan kodu yürütmektedirler. Semafor X=2 ve semafor Y =0 ilk değerleri ile yaratılmışlardır. Sistemde A tipinden 3 adet, B tipinden de 2 adet örnek proses canlandırılmıştır.

|  |  |
| --- | --- |
| Proses A | Proses B |
| P(X)  V(Y) | 1.P(Y)  2.P(Y)  3.V(X)  4.V(Y) |

a) Proseslerin **AABAB** sıralamasında sonlanmaları mümkün müdür? Yanıtınızı, bu sonucu yaratacak bir yürütme sekansı örneği verip, semaforların alacakları değerleri de göstererek açıklayın.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A1 | A2 | A3 | B1 | B2 | Sem X | Sem Y |
|  |  |  |  |  | 2 | 0 |
| P(X) |  |  |  |  | 1 |  |
|  | p(x) |  |  |  | 0 |  |
|  |  |  | 1.p(y) bloke |  |  | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | p(x) bloke |  |  | 0 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| v(y) ->b1 |  |  | 2.p(y) bloke |  |  | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v(y) ->b1 |  | 3->v(x)-a3 uyan |  |  | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | v(y) |  |  |  | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 1. p(y) |  | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 2. p(y) bloke |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 4.v(y) b3 uyandir | 3. v(x) | 1 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 4.v(y) |  | 1 |

Sonlanir.

b) Proseslerin **AABBA** sıralamasında sonlanmaları mümkün müdür? Yanıtınızı, bu sonucu yaratacak bir yürütme sekansı örneği verip, semaforların alacakları değerleri de göstererek açıklayın.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A1 | A2 | A3 | B1 | B2 | Sem X | Sem Y |
|  |  |  |  |  | 2 | 0 |
| p(x) |  |  |  |  | 1 |  |
|  | p(x) |  |  |  | 0 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 1.p(y) bloke |  |  | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 1.p(y) bloke |  | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | p(x) bloke |  |  | 0 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| v(y) -> b1 uyandir |  |  | 2.p(y)  bloke |  |  | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v(y) ->b2 uyandir |  |  | 2.p(y) bloke |  | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |

Hayir sonlanmaz, vy degeri hep uyandirma islemi yaptigindan semafor degeri artmayacak, a3px te hic vx cagrisi getirmedigi icin hicbir zaman sonlanmayacak.

**Soru 3: (20 puan)**

A,B,C ve D proseslerine ait yürütme planı aşağıda gösterilmiştir: A prosesi tamamlanmadan B ve C başlayamazlar, D ise B ve C tamamlanmadan başlayamaz. Bu çalışma düzenin gerçekleyecek şekilde, mesaj aktarımı ilkelleri ve posta kutusu kullanarak her prosese ait kodu yazın. İletişimin türünü (senkron/asenkron) belirtin.

****

mesaj = “bos mesaj”;

n=2 tampon; //2 kanal var cunku istersek cogaltiri

create\_mb(uret); //bos bos zarf yarattik

create\_mb(tuket); //dolu // a da uretilen dolu zarflari aliyoruz

create\_mb(bitti); // b ve c’den gelenleri receive ile blocking sekilde bekliyoruz

for (i = 0 ; i<n; i++) send(uret,mesaj);

Process A /// 2 tane mesaj uretti blocking receive nonblocking send

while(true) {

receive(uret,mesaj)

mesaj = “yeni veri”

send(tuket,mesaj)

}

Process B{ // A’dan gelen mesaji aldi, ve bos zarf uretti // blocking receive nonblocking send

receive(tuket,mesaj)

// mesaji kullandi

send(bitti,mesaj)

}

Process C

{ // A’dan gelen mesaji aldi, ve bos zarf uretti blocking receive nonblocking send

receive(tuket,mesaj)

// mesaji kullandi

send(bitti,mesaj)

}

Process D // blocking receive // send yok

while(true) {

receive(bitti,mesaj)

// mesaj kullan

//herhangi bir send yapmasina gerek yok

}

--- ASENKRON iletisim yapisi

**Soru 4: (20 puan)**

Bir bağlantılı liste yapısı üzerinde yürütecekleri işlem türüne göre sınıflandırılan üç tip proses mevcuttur. **Arama** işlemi gerçekleştiren *arayıcı prosesler* belirli bir değerin listede yer alıp almadığına araştırırlar; çok sayıda arayıcı proses paralel işlem görebilir. **Ekleme** işlemi gerçekleştiren *ekleyici prosesler* listenin sonuna yeni bir değer eklerler. Ekleyici prosesler birbirlerini dışlayarak çalışmalıdırlar, ancak ekleme işlemi çok sayıdaki arama işlemi ile birarada sürdürülebilir. **Silme** işlemi gerçekleştiren *silici prosesler* ise listenin herhangi bir yerinden bir elemanı silerler. Silme işlemi diğer işlemlerden ayrık olarak yürütülmelidir.

**Semafor** yapısını kullanarak, tanımlanan tiplerden proseslerin bu çalışma düzenine uygun şekilde işlemlerini yerine getirmelerine olanak sağlayacak olan kodu yazın, **semaforların ilk değerlerini** belirtin. Söz konusu liste işlemlerini yerine getiren fonksiyonların var olduğunu kabul edebilirsiniz.

**monitor linked\_list {**

**int n\_ara, n\_bekle, n\_sil;**

**condition ara\_bekle, ekle\_bekle, sil\_bekle;**

**//Arama işlemi gerçekleştiren arayıcı prosesler belirli bir değerin listede yer alıp almadığına araştırırlar; çok sayıda arayıcı proses paralel işlem görebilir.**

**entry ara\_basla**

**begin**

**if (n\_sil > 0) {**

**cwait(ara\_bekle);// kuyruga aldik**

**}**

**n\_ara := n\_ara + 1; // ilk buraya gelip arttirdi.**

**c\_signal(ara\_bekle);**

**end**

**entry ara\_bitir**

**begin**

**n\_ara := n\_ara - 1;**

**if (n\_ara == 0) and (n\_ekle == 0) {**

**c\_signal(sil\_bekle); //bekleyen yoksa geri don**

**}**

**end**

**//Ekleme işlemi gerçekleştiren ekleyici prosesler listenin sonuna yeni bir değer eklerler.**

**entry ekle\_basla**

**begin**

**if (n\_ekle > 0) or (n\_sil > 0) {**

**c\_wait(ekle\_bekle); //kuyruga al**

**}**

**n\_ekle := n\_ekle + 1;**

**end**

**entry ekle\_bitir**

**begin**

**n\_ekle := n\_ekle - 1;**

**c\_signal(ekle\_bekle);**

**if (n\_ara == 0) and (n\_ekle == 0) {**

**c\_signal(sil\_bekle) // uyandir**

**}**

**end**

**//Silme işlemi gerçekleştiren silici prosesler ise listenin herhangi bir yerinden bir elemanı silerler. Silme işlemi diğer işlemlerden ayrık olarak yürütülmelidir.**

**entry sil\_basla**

**begin**

**if (n\_ara > 0) or (n\_ekle > 0) or (n\_sil > 0) {**

**c\_wait(sil\_bekle) // kuyruukkkk**

**}**

**n\_sil := n\_sil + 1;**

**end**

**entry sil\_bitir**

**begin**

**n\_sil := n\_sil - 1;**

**c\_signal(ara\_bekle);**

**c\_signal(ekle\_bekle);**

**if (n\_ara == 0) and (n\_ekle == 0) {**

**c\_signal(sil\_bekle);**

**}**

**end**

**{ // ilk degerleri**

**n\_ara := 0;**

**n\_ekle := 0;**

**n\_sil := 0;**

**}**

**}**

**Soru 5: (20 puan)**

**“**m” adet üretici ve “n” adet tüketici prosesin yer aldığı bir ortamda, bir üretici tamsayı tipinden “v” verisini tüm tüketicilere iletmek üzere ***yayınla (v)*** çağrısını kullanmaktadır. Her tüketici ***veriyi\_al***çağrısını yürüterek yayınlanan verinin bir kopyasını elde edebilmektedir. Tüketici bir veriyi sadece bir kez elde edebilmeli, yeni bir veri hazir olmadığı taktirde ***veriyi\_al***çağrısından yeni bir veri gelene kadar geri dönememelidir. Bu haberleşme düzenini gerçekleyen bir **monitör** tasarlayın. Monitör sadece **bir adet** veriyi tamponlamalıdır, öyle ki, bir üreticinin yayınla çağrısını izleyen bir diğer üreticinin yayınla çağrısı, n adet tüketicinin tümü ilk çağrıya ilişkin verinin bir kopyasını elde edene kadar bekletilmelidir. **Koşul değişkenlerini** ne amaçla kullandığınız ayrıca belirtin.

monitor veri\_yayinla {

int counter;

bool veri\_var;

bool okudum[1..n];

int tampon;

cond herkes\_aldi;

cond veri\_hazir;

empty yayinla (veri : int)

begin

if (veri\_var)

then c\_wait(herkes\_aldi);

tampon := veri;

veri\_var = true;

csignal(veri\_hazir);

end

empty veriyi\_al (t : int, i = process\_id)

begin

if (okudum[i] or ((not)veri\_var)) // Tüketici bir veriyi sadece bir kez elde edebilmeli

then c\_wait(veri\_hazir); // bekleme kuyruguna al

counter = counter +1;

t :=tampon;

okudum[i] : = true // oldugum process degerini okudum olarak isaretle

if(sayac == n) then sayac // sona geldik

begin

counter:= 0;

veri\_var :=false;

c\_signal(herkes\_aldi); // herkes aldi uyandir yok ise geri don

end

c\_signal(veri\_hazir); // sadece bir tuketici uyandir

end

begin // baslangic degerleri

veri\_var := false;

counter :=0;

for i =1 to n do // okudum degerlerinin hepsini false a cekiyoruz

okudum[i] : = false;

end

}

Kosul degiskenlerini, process bloke olmadan once monitoru terk etmelerini saglayan ozel bir yapi saglamak amaciyla olusturulmustur. bloke olan process diger proseslerin monitore erismelerine engel olmadan, monitor disinda, kosullarin elverisli hale gelmesini bekleyebilir. Hem karsilikli dislama hem de senkronizasyonu sagliyor buradaki kosul degiskenlerimiz.