SO Curs Extra

Sincronizarea proceselor

Concurenta

Acces concurent
Conditii de cursa
Atomicitate
Regiuni critice
Excludere mutuala

Acces exclusiv

Dezactivarea intreruperilor
Kernel preemptiv
Lock-uri/Mutex-uri
Hardware de sincronizare
Spinlock-uri
Spinlock-uri vs. mutex-uri

Comunicarea între procese

Comunicare
Concurenta/competitie
Colaborare
Acces exclusiv
Sincronizare

Sincronizare

Sincronizare/asteptare
Semafoare
Problema producator/consumator
Problema cititori-scriitori

Problematica sincronizarii

Deadlock-uri
Inconsecventa datelor
Overhead de sincronizare

Atomicitate

Conditii de cursa

Regiuni critice

Excludere mutuala

Acces concurent

Concurenta

Colaborare

Comunicare

Colaborare
Concurenta/competitie
Coordonare/secventializare

Sincronizare

Cuprins

Suport de curs

Lock-uri, mutex-uri

Kernel preemptiv

Hardware de sincronizare

Dezactivarea intreruperilor

Spinlock

Operatii cu semafoare

Spinlock vs. mutex

Semafoare

Problema producator-consumator

Sincronizarea operatiilor

Problema cititori-scriitori

Problema cititori-scriitori (2)

Cuvinte cheie

Overhead de sincronizare

Modelarea deadlock-urilor

Deadlock-uri

Provenirea, evitarea deadlock-urilor

Inconsecventa datelor

Prezi

SO Curs Extra

Sincronizarea proceselor

Concurenta

Acces concurent
Conditii de cursa
Atomicitate
Regiuni critice
Excludere mutuala

Acces exclusiv

Dezactivarea intreruperilor
Kernel preemptiv
Lock-uri/Mutex-uri
Hardware de sincronizare
Spinlock-uri
Spinlock-uri vs. mutex-uri

Comunicarea între procese

Comunicare
Concurenta/competitie
Colaborare
Acces exclusiv
Sincronizare

Sincronizare

Sincronizare/asteptare
Semafoare
Problema producator/consumator
Problema cititori-scriitori

Problematica sincronizarii

Deadlock-uri
Inconsecventa datelor
Overhead de sincronizare

Atomicitate

Conditii de cursa

Regiuni critice

Excludere mutuala

Acces concurent

Concurenta

Colaborare

Comunicare

Colaborare
Concurenta/competitie
Coordonare/secventializare

Sincronizare

Cuprins

Suport de curs

Lock-uri, mutex-uri

Kernel preemptiv

Hardware de sincronizare

Dezactivarea intreruperilor

Spinlock

Operatii cu semafoare

Spinlock vs. mutex

Semafoare

Problema producator-consumator

Sincronizarea operatiilor

Problema cititori-scriitori

Problema cititori-scriitori (2)

Cuvinte cheie

Overhead de sincronizare

Modelarea deadlock-urilor

Deadlock-uri

Provenirea, evitarea deadlock-urilor

Inconsecventa datelor

Prezi

Suport de curs

OSCE
- Capitolul 6 – Process Synchronization
  - Sectiunile 6.1-6.5, 6.6.1, 6.6.2, 6.8.2-6.8.4
- Capitolul 7 - Deadlocks

MOS
- Capitolul 2 – Processes and Threads
  - Sectiunile 2.3.1–2.3.6, 2.3.9, 2.4.2
- Capitolul 3 - Deadlocks

Little Book of Semaphores
- Capitolele 1, 2, 3
- Capitolul 4 – Sectiunile 4.1, 4.2

Prezi

Cuprins

Prezi

Comunicarea între procese

Comunicare
Concurenta/competitie
Colaborare
Acces exclusiv
Sincronizare

Prezi

Comunicare

Colaborare

Concurenta/competitie

Coordonare/secventializare

Prezi

Concurenta

rezultate predictibile
accesul concurent poate produce date incoerente
serializarea accesului

Prezi

Colaborare

schimb de informatie
partajarea informatiei

Prezi

Acces exclusiv

Un proces (P1) foloseste resursa R
Un alt proces (P2) solicită resursa R prelucrată de P1
P2 trebuie să astepte eliberarea resursei R

Prezi

Sincronizare

P1 foloseste R1 si produce R2
P2 solicită R2
P2 trebuie să astepte ca P1 să producă R2

Prezi

Concurenta

Acces concurent
Conditii de cursa
Atomicitate
Regiuni critice
Excludere mutuala

Prezi

Acces concurent

Resursă comună – fisier, zonă de memorie

Situatie

```
a = 0 /* initializare */
```

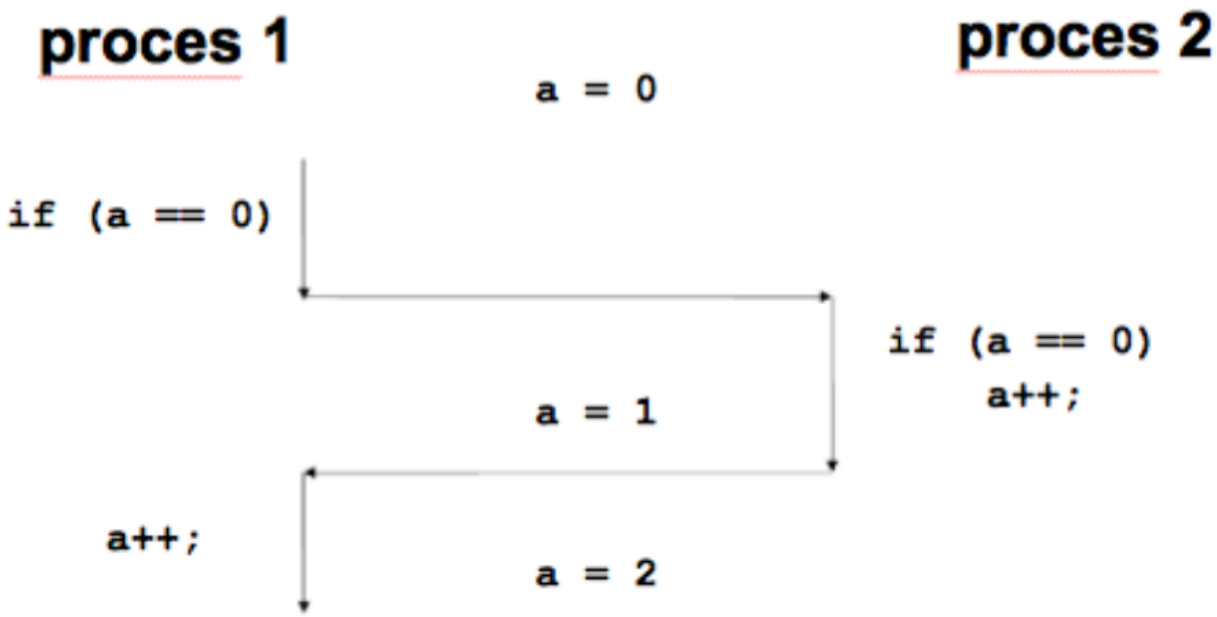
două instante de executie rulează:

```
if (a == 0)
    a++;
```

Ce valoare va avea a după executie?

Prezi

Conditii de cursa



Prezi

Atomicitate

Operaii care se execută într-un singur ciclu de instruciune
În câte cicluri se execută a++?

thread1	thread2	a	reg1	reg2
load a, reg1		0	0	0
inc reg1		0	1	0
	load a, reg2	0	1	0
	inc reg2	0	1	1
	store reg2, a	1	1	1
store reg1, a		1	1	1

Prezi

Regiuni critice

Critical sections

parti din program in care sunt accesate resurse partajate
pot fi executate in paralel de mai multe instante de executie

a++
parcurgerea/adaugarea/stergerea unor elemente dintr-o lista

se pot genera conditii de cursa

protejare prin excludere mutuala
- o singura instanta de executie are acces la regiunea critica
- serializarea accesului

Prezi

Excludere mutuala

Mutual exclusion

acces exclusiv

un singur proces are acces la regiunea critica

locking

busy waiting vs. blocking

Prezi

Sincronizare

Sincronizare/secventiere
Semafoare
Problema producator/consumator
Problema cititori-scriitori

Prezi

Sincronizarea operatiilor

Secventiere, succesiune, ordonare

sincronizare "pura"

un proces asteapta alt proces

o actiune depinde de alta actiune

sincronizarea accesului = ordonarea operatiilor

- se garanteaza o secventa, succesiune

Prezi

Semafoare

primitive de sincronizare, ordonare

un proces sau mai multe asteapta dupa alt proces

intern este un contor

contor = 0 -> actiune neterminata
contr > 0 -> N actiuni terminate

Prezi

Operatii cu semafoare

up - incrementarea contorului
- actiune terminata
- post, signal, V

down - decrementarea contorului
- asteptarea unei actiuni
- blocanta daca valoarea contorului este 0 (zero)
- get, wait, P

Prezi

Problema producator-consumator

```
semaphore to_full = N;
semaphore to_empty = 0;
mutex_t m;
```

```
void consumer(void)
{
    int item;
    while(TRUE) {
        down(to_empty);
        lock(mutex);
        item = remove_item();
        unlock(mutex);
        up(to_full);
        consume_item(item);
    }
}
```

```
void producer(void)
{
    int item;
    while (TRUE) {
        item = produce_item();
        down(to_full);
        lock(mutex);
        insert_item(item);
        unlock(mutex);
        up(to_empty);
    }
}
```

Prezi

Problema cititori-scriitori

```
void writer(void)
{
        down(excl);
        ...
        /* do writing */
        ...
        up(excl);
}
```

```
void reader(void)
{
        lock(mutex);
        read_count++;
        if (read_count == 1)
                down(excl);
        unlock(mutex);
        ...
        /* do reading */
        ...
        lock(mutex)
        read_count--;
        if (read_count == 0)
                up(excl);
        unlock(mutex);
}
```

Prezi

Problema cititori-scriitori (2)

Neajunsuri

posibil starvation la scriitori
- multi cititori

prioritate pe scriitori
- daca e cel putin un scriitor prezent, cititorii nu se executa

de ce e putin probabil starvation la cititori?

Prezi

Acces exclusiv

Dezactivarea întreruperilor
Kernel preemptiv
Lock-uri/Mutex-uri
Hardware de sincronizare
Spinlock-uri
Spinlock-uri vs. mutex-uri

Prezi

Dezactivarea intreruperilor

De ce apare acces concurent?

- un proces intrerupe pe altul

Solutia simpla: dezactiveaza planificatorul

dezactivarea intreruperilor

- intreruperea de ceas
- expirarea cuantei sau proces prioritar nu cauzeaza context switch

Prezi

Kernel preemptiv

cod kernel comun tuturor proceselor
- susceptibil la conditii de cursa

preemptiv
- proces in timp ce ruleaza cod kernel
- mai complex de implementat
- timp bun de raspuns

nepreemptiv
- un proces e preemptat doar la iesirea din kernel
- mai simplu de implementat
- latenta mai mare
- in continuare probleme pe sisteme SMP

Prezi

Lock-uri, mutex-uri

Acces exclusiv

doua stari: locked, unlocked

procesele se blocheaza in starea locked

operatii de lock si unlock
- acquire, release

un mutex poate fi implementat ca un semafor binar

Prezi

Hardware de sincronizare

TestAndSet/TSL (Test and Set Lock)

- compare & exchange în alte denumiri (cmpxchg)

TSL RX, LOCK

- se citeste variabila LOCK in registrul RX
- se pune valoarea TRUE (activat) în LOCK
- operatia se executa atomic

```
boolean TestAndSet(boolean *target)
{
    boolean rv = *target;
    *target = TRUE;
    return rv;
}
```

Prezi

Spinlock

lock/unlock cu busy waiting

necesită suport hardware

```
enter_region:
    TSL RX, LOCK
    CMP RX, 0
    JNE enter_region
```

```
exit_region:
    MOV LOCK, 0
```

Prezi

Spinlock vs. mutex

- consum timp de procesor
+ rapid, overhead redus la lock/unlock

pentru regiuni critice scurte

+ nu consuma timp in lock
- overhead pe lock

pentru regiuni critice mari
pentru regiuni critice cu I/O

Prezi

Problematica sincronizarii

Deadlock-uri
Inconsecventa datelor
Overhead de sincronizare

Prezi

Deadlock-uri

asteptare simultana a doua sau mai multe procese

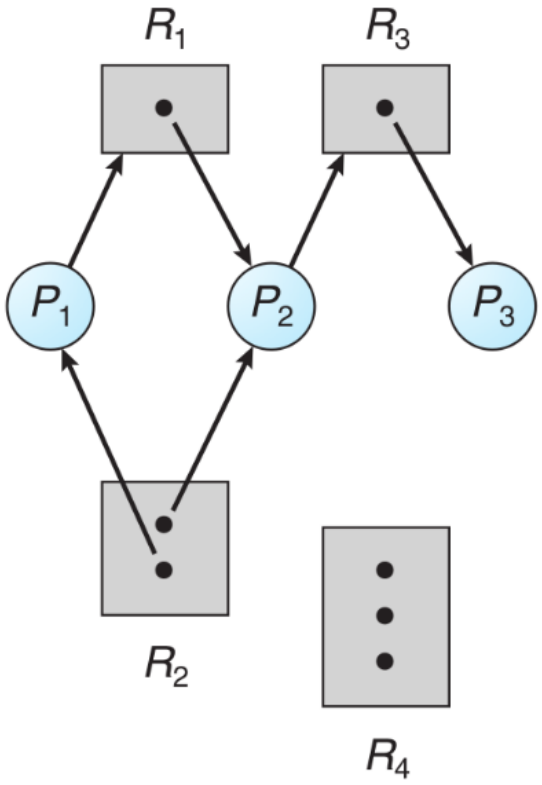
conditii

- excludere mutuala
- hold & wait
- fara preemptare
- asteptare circulara

Prezi

Modelarea deadlock-urilor

graf de alocare a resurselor



Prezi

Prevenirea, evitarea deadlock-urilor

detectare deadlock-uri: cicluri in graf si procese blocate

obtinerea lock-urilor in aceeasi ordine

obtinerea tuturor lock-urilor in acelasi timp
- ineficienta, serializare

Prezi

Inconsecventa datelor

absenta sincronizarii

acces exclusiv nepotrivit

- doua operatii atomice succesive nu inseamna operatie atomica in ansamblu
- pierderea notificarii pentru sincronizare

Prezi

Overhead de sincronizare

contention rate: numarul de procese care vor sa faca lock

starvation: doar unele procese intra in regiunea critica

thundering herd: se trezesc toate procesele din lock

apel de sistem, invocarea scheduler-ului, context switch

Prezi

Cuvinte cheie

§

concurenta
colaborare
acces exclusiv
sincronizare
excludere mutuala
atomicitate
conditie de cursa
regiune critică
mutex

TSL
spinlock
semafor
producator-consumator
cititori-scriitori
deadlock
graf de alocare a resurselor
inconsecventa datelor
overhead

Prezi