Universiteti i Prishtinës "Hasan Prishtina"

Fakulteti Inxhinierisë Elektrike dhe Kompjuterike

Dokumentim teknik i projektit

Lënda: Sisteme Operative

Detyra : Të implementohet programi për sinkronizimin e proceseve përmes problemit klasik bounded-buffer. Të diskutohen zgjidhjet që i ofron ky model, si dhe situatat të cilat nuk mund t'i zgjidhë.

Emri profesorit/Asistentit	**Emri & mbiemri studentëve / email adresa**	
Prof. Artan Mazrekaj	1. Diellza Përvetica	diellza.perveticai@student.uni-pr.edu
Ass. Dalinë Vranovci	2. Donjeta Morina	donjeta.morina9@student.uni-pr.edu

Prishtinë, 2022

Përmbajtja

Abstrakti 3

I. Hyrje 3

II. Përshkrimi i problemit 4

III. Teknologjia 5

IV. Implementimi 11

4.1 Semaforët dhe llojet 11

4.1.1. Semaforët binar 11

4.1.2. Semaforët counting dhe empty 12

4.2. Operacionet e Prodhuesit dhe Konsumatorit 12

V. Përfundimi 13

Tabela e figurave 14

Referencat 14

Abstrakti

Me anë të këtij punimi do të mundohemi të tregojmë se si një prodhues përpiqet të fus të dhëna në një vend të zbrazët buffer –in. Bounded-Buffer është një shembull klasik i aksesit të njëkohshëm në një burim të përbashkët i cili mund të thuhet se lejon që prodhues të shumtë dhe konsumatorë të shumtë të ndajnë një Buffer të përbashkët dhe në këtë dokumentim është e analizuar shumë mirë pjesa e konsumatorit dhe prodhuesit.

I. Hyrje

Një buffer me madhësi fikse shërben si radhë si për prodhuesin ashtu edhe për konsumatorin. Përgjegjësia e prodhuesit është të gjenerojë të dhëna dhe t'i ruajë ato në buffer. Është përgjegjësi e konsumatorit që të konsumojë të dhënat nga ky buffer një nga një.

Si siguroheni që prodhuesi të mos përpiqet të vendosë të dhëna në buffer kur ai është plot dhe që konsumatori të mos përpiqet të konsumojë të dhëna kur buferi është bosh?

Kur një prodhues përpiqet të vendosë të dhëna në një buffer që tashmë është i mbushur, ai harxhon ciklet e CPU-së. E njëjta gjë vlen edhe për konsumatorin, i cili përpiqet të konsumojë nga një buffer bosh. Në këto raste, preferohet që ata të shkojnë në gjumë në mënyrë që programuesi të planifikojë një proces tjetër.

II. Përshkrimi i problemit

Problemi i buffer-it bounded është një shembull i njohur i aksesit të njëkohshëm në një burim të përbashkët (i njohur gjithashtu si problemi prodhues-konsumator). Një bounded buffer lejon shumë prodhues dhe konsumatorë të ndajnë të njëjtin bounded buffer. Prodhuesit shkruajnë të dhëna në buffer, ndërsa konsumatorët lexojnë të dhëna prej tij.

Problemi i buferit të kufizuar përfshin një buffer me n slota, secila prej të cilave mund të ruajë një njësi të dhënash. Dy proceset që funksionojnë në Buffer janë Prodhuesi dhe Konsumatori. Prodhuesi përpiqet të fusë të dhëna në një vend të zbrazët, ndërsa konsumatori përpiqet t'i heqë ato. Kur proceset drejtohen njëkohësisht, rezultatet e pritura nuk do të merren.Si prodhuesi ashtu edhe konsumatori duhet të funksionojnë në mënyrë të pavarur.

III. Teknologjia

Në këtë projekt gjuha e përdorur është C dhe analizimi i të gjithë kodit bëhet në gjuhën C. Ku si fillim do të analizojm mënyrën e incializimit figura 1.

```
// Incializimi

Mutex mutex; //
Semophore empty = N; // sefarori i zbresast, pasi qe madhesia eshte N
Semophore full = 0;  // i mbushur
int in = 0;    // per itemmet e perdora
int out = 0;  // per cilat
int buffer [N];
```

Figura 1: Inicializimi

Pas përfundimit të inicializimit do analizojm dhe krijojmë kodin për prodhuesit dhe konsumatore. Të gjitha të lidhura me kodin paraprak të inicializimit. Së pari paraqesim pjesen e prodhuesit në figura 2. Ku është krijuar një unazë dhe janë analizuar prit dhe sinjali i prodhuesit për të analizuar nëse konsumtori mundë të e përdor.

```
while (True) {
   wait(empty); //pret ose ka qetesi derisa ka hapsire
   wait(mutex);
   buffer[in] = item //merr disa item
   in = (in+1)% buffersize;
   signal(mutex);
   signal(full); //  sinjali te konsumatorit  qe gjendet ne te dhenat dhe ta perdor
}
```

Figura 2: Prodhuesi

Pastaj dhe vazhdojm te pjesa e konsumatorit ku lidhur me pjesen e prodhuesit analizoj se sa kohë mundë të fusim prodhuesit tjera kjo paraqitet në figura 3.

```
//konsumatori
while(True) {
   wait(full); // pret
   wait(mutex);
   item = buffer[out];
   out = (out+1)%buffersize;
   signal(mutex);
   signal(empty); // e analizon nese eshte zbrazet dhe lejon qe te shtone
 }
```

Figura 3: Konsumatori

Dhe kodi i përgjithshëm për shfaqjen e prodhuesi dhe konsumatorëve është i paraqitur në figura 4, figura 5 dhe figura 6.

```c
//librarit e inicializuar
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>

/*
Janë përdorur 5 produkte dhe 5 konsumator
*/

#define MaxItems 5
#define BufferSize 5 // madhesia e buffersize

sem_t empty; //deklerimi i semaforeve
sem_t full;
int in = 0;
int out = 0;
int buffer[BufferSize];
pthread_mutex_t mutex; //deklarimi i nje mutex
// është një semafor binar me një kufizim pronësie
// dhe ofron një mbrojtje disi më të fortë se
//një semafor i zakonshëm
```

Figura 4: Kodi i pergjithshem 1

```c
void *producer(void *pno)
{
    int item;
    for(int i = 0; i < MaxItems; i++) {
        item = rand(); //nje produkt te randomt
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = item;
        printf("Producer %d: Insert Item %d at %d\n", *((int *)pno),buffer[in],in);
        in = (in+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
}
void *consumer(void *cno)
{
    for(int i = 0; i < MaxItems; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
        printf("Consumer %d: Remove Item %d from %d\n",*((int *)cno),item, out);
        out = (out+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}
```

Figura 5: Kodi i përgjithshëm 2

```c
int main()
{

    pthread_t pro[5],con[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty,0,BufferSize);
    sem_init(&full,0,0);

    int a[5] = {1,2,3,4,5}; //per numeri te produkit dhe konsumatorit
    for(int i = 0; i < 5; i++) {
        pthread_create(&pro[i], NULL, (void *)producer, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&con[i], NULL, (void *)consumer, (void *)&a[i]);
    }

    for(int i = 0; i < 5; i++) {
        pthread_join(pro[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(con[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);

    return 0;

}
```

Figura 6: Kodi i përgjithshëm 3

Pasi ky kodë bëhet run dhe kompailohet athër në kompailim do na paraqiten prodhuesit e marra rasisht dhe konsumatorët, ku janë të paraqitur si në vazhdim :

Producer 5: Insert Item 1804289383 at 0

Producer 5: Insert Item 1681692777 at 1

Producer 5: Insert Item 1714636915 at 2

Producer 5: Insert Item 1957747793 at 3

Consumer 3: Remove Item 1804289383 from 0

Consumer 3: Remove Item 1681692777 from 1

Consumer 5: Remove Item 1714636915 from 2

Producer 1: Insert Item 846930886 at 4

Producer 1: Insert Item 719885386 at 0

Producer 1: Insert Item 1649760492 at 1

Producer 1: Insert Item 596516649 at 2

Consumer 4: Remove Item 1957747793 from 3

Consumer 4: Remove Item 846930886 from 4

Consumer 5: Remove Item 719885386 from 0

Consumer 1: Remove Item 1649760492 from 1

Consumer 3: Remove Item 596516649 from 2

Producer 5: Insert Item 424238335 at 3

Consumer 4: Remove Item 424238335 from 3

Producer 1: Insert Item 1189641421 at 4

Consumer 5: Remove Item 1189641421 from 4

Producer 4: Insert Item 1025202362 at 0

Producer 4: Insert Item 1350490027 at 1

Producer 4: Insert Item 783368690 at 2

Producer 4: Insert Item 1102520059 at 3

Producer 4: Insert Item 2044897763 at 4

Consumer 5: Remove Item 1025202362 from 0

Consumer 4: Remove Item 1350490027 from 1

Consumer 2: Remove Item 783368690 from 2

Consumer 3: Remove Item 1102520059 from 3

Producer 3: Insert Item 1967513926 at 0

Producer 3: Insert Item 1540383426 at 1

Producer 3: Insert Item 304089172 at 2

Consumer 3: Remove Item 2044897763 from 4

Producer 3: Insert Item 1303455736 at 3

Consumer 4: Remove Item 1967513926 from 0

Producer 3: Insert Item 35005211 at 4

Consumer 5: Remove Item 1540383426 from 1

IV. Implementimi

Për të implementuar një array me madhësi të kufizuar në memorie ndahet nga një numër thread-ash prodhues dhe konsumator për të zbatuar bufferin e kufizuar. Threads të prodhuesit "prodhojnë" një item dhe e pozicionojnë item-in në grup. Threads të konsumatorit heqin dhe "konsumojnë" një item nga array.

Struktura përmban një vlerë grupi për ruajtjen e vlerave të numrave të plotë në buffer, si dhe dy indekse të numrave të plot next_in dhe next_out. Indeksi next_in përdoret për të mbajtur gjurmët se ku duhet të shkruhet artikulli next_data në buffer. Indeksi në vijim mban gjurmët se ku mund të lexohet item next_data nga buferi.

Tre elementë të të dhënave, B, C dhe D, janë aktualisht në biffer në shembullin e mëposhtëm. Të dhënat do të shkruhen për të indeksuar next_in = 4 në shkrimin tjetër. Të dhënat do të lexohen nga indeksi next_out = 0 në next_read. Figura 7 paraqitja e Bounded Buffer.

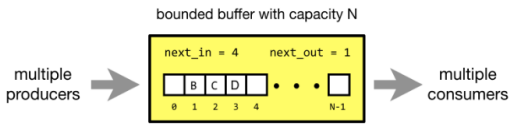

Figura 7: Bounded Buffer

4.1 Semaforët dhe llojet

Semaforët janë një zgjidhje për problemin e bounded buffer të kufizuar. Përdoren semaforët e mëposhtëm:

1. m, a Binary Semaphore.
2. empty, a Counting Semaphore.
3. full, a Counting Semaphore.

4.1.1. Semaforët binar

Të gjitha përditësimet e gjendjes së buffer-it duhet të kryhen në një seksion kritik. Më konkretisht, përjashtimi i ndërsjellë duhet të zbatohet ndërmjet seksioneve kritike të listuara më poshtë:
- Një prodhues writes në një slot të buffer-it dhe update-on next_in.
- Një konsumer që lexon nga sloti i buffer-it dhe update-on next_out.
Për të kufizuar aksesin në seksionet kritike, mund të përdoret një semafor binar.

4.1.2. Semaforët counting dhe empty

Nëse buferi është plot, prodhuesit duhet të bllokojnë. Nëse buferi është bosh, konsumatorët duhet të bllokojnë. Kjo mund të arrihet duke përdorur dy semaforë counting.

Për të numëruar hapësirat boshe në buffer, përdoret një semafor të quajtur empty. Vendoset vlera e këtij semafori në N, përpara se të shkruajmë në buffer, një prodhues duhet të presë në këtë semafor. Pas leximit nga buferi, një konsumator do të sinjalizojë këtë semafor. Për të numëruar numrin e items të të dhënave në buffer, përdoret një semafor të quajtur të dhëna. Vendoset vlera e këtij semafori në 0. Para se të lexojë nga buferi, një konsumator duhet të presë në këtë semafor. Pas shkrimit në buffer, një prodhues do të sinjalizojë këtë semafor.

4.2. Operacionet e Prodhuesit dhe Konsumatorit

Operacionet e Prodhuesit në figura 8

```
do {
  wait(empty); // wait until empty>0 and then decrement 'empty'
  wait(mutex); // acquire lock
  /* perform the
  insert operation in a slot */
  signal(mutex); // release lock
  signal(full);  // increment 'full'
} while(TRUE)
```

Figura 8: Operacionet e përgjithshme të prodhuesit

Mund të shohim nga kodi i mësipërm për një prodhues që një prodhues fillimisht pret derisa të ketë të paktën një vend të zbrazët. Më pas, semafori i zbrazët zvogëlohet sepse tani do të ketë një vend të zbrazët më pak sepse prodhuesi do të fusë të dhëna në një nga ato slota. Pastaj ai fiton një bllokim në buffer, duke e penguar konsumatorin të hyjë në të derisa prodhuesi të përfundojë funksionimin e tij. Për shkak se prodhuesi sapo ka mbushur një slot në buffer, bllokimi vendoset të lirohet dhe vlera e plotë rritet pas operacionit të futjes.

Operacionet e Konsumatorit figura 9

```
do {
  wait(full); // wait until full>0 and then decrement 'full'
  wait(mutex);  // acquire the lock
  /* perform the remove operation
   in a slot */
  signal(mutex); // release the lock
  signal(empty); // increment 'empty'
} while(TRUE);
```

Figura 9: Operacionet e përgjithshme të kosumatorit

Mund të shohim nga kodi i mësipërm Konsumatori pret derisa buferi të ketë të paktën një slot të plotë. Pasi konsumatori përfundon funksionimin e tij, ai zvogëlon të gjithë semaforin sepse numri i sloteve të zëna zvogëlohet me një. Pas kësaj, konsumatori merr një bllokues në buffer. Konsumatori më pas përfundon operacionin e heqjes, duke hequr të dhënat nga një nga slotet e plota. Konsumatori më pas zhbllokon bllokimin. Së fundi, për shkak se konsumatori sapo ka hequr të dhënat nga një vend i zënë, semafori i zbrazët rritet me një.

V. Përfundimi

Prodhuesi : Kur ka të paktën një slot bosh, operacioni i prodhuesit zvogëlon Semaforin sepse do të ketë një slot më pak bosh. Të dhënat më pas vendosen në slotin bosh. Si rezultat, buffer bllokohet. Sapo futen të dhënat, vlera e plotë rritet për të treguar që prodhuesi ka mbushur një slot.

Tabela e figurave

Figura 1: Inicializimi 5
Figura 2: Prodhuesi 5
Figura 4: Konsumatori 5
Figura 5: Kodi i pergjithshem 1 6
Figura 6: Kodi i përgjithshëm 2 7
Figura 7: Kodi i përgjithshëm 3 8
Figura 8: Bounded Buffer 11
Figura 9: Operacionet e përgjithshme të prodhuesit 12
Figura 10: Operacionet e përgjithshme të kosumatorit 12

Referencat

[1] os-introduction: i2Tutorials," i2Tutorials, 2019. [Online]. Available: https://www.i2tutorials.com/os-introduction/os-classical-problems-of-synchronization/. [Accessed 18 November 2022]

[2] Sandeep Jain, Aditya Save, "Producer-Consumer: GeeksforGeeks," GeeksforGeeks, 2009. [Online]. Available: https://www.geeksforgeeks.org/producer-consumer-problem-in-c/. [Accessed 19 November 2022]

[3] Sandeep Jain, Aditya Save, "OpenMP: GeeksforGeeks," GeeksforGeeks, 2009. [Online]. Available: https://www.geeksforgeeks.org/openmp-introduction-with-installation-guide/. [Accessed 19 November 2022].

[4] Operating Systems," Uppsala University, 2018. [Online]. Available: http://www.it.uu.se/education/course/homepage/os/vt18/. [Accessed 18 November 2022]