Opertaing Systems 2 Project

Bound-Buffer problem

Members:

|  |  |
| --- | --- |
| ID | Name |
| 201900016 | أحمد أشرف عبد المنعم محمود |
| 202000613 | عمر ناصر على سيد |
| 202000808 | محمد عماد عبد القادر حنفي |
| 202000248 | حازم طارق رجب محمد |
| 202000471 | صموئيل سمير نسيم كامل |
| 202000537 | عبدالرحمن مصطفى احمد محمد |

Pseudocode:

//This program solves bounded buffer (producer consumer) problem

//Our program consists of 4 classes : main, buffer, producer, consumer

//1) class

Class Buffer{

List buffer

Capacity := 5 //assume that buffer size is 5

Value := 0

// Function called by producer thread

Function produceData () {

While true{

Synchronized (instance){

// prducer thread waits while buffer

// is full

while (buffer size = capacity) {

wait()

}

// to insert the data unit in the buffer

buffer add(value)

value := value + 1

// notifies the consumer thread that

// now it can start consuming data unit

notify()}

}

// Function called by Customer thread

Function consumeData() {

//assuming consumer will consume 10 data units only.

While numberof data<10 { //or while(true)

synchronized (instance) {

// producer thread waits while buffer

// is empty

while (buffer size = 0) {

wait()

}

// consumer retrieve the first delivered existed data unit in the buffer

int val := buffer removeFirst()

// Wake up producer thread

notify()

}

}

}

//2) class

class Producer {

buffer //create an object of buffer

Producer(Buffer passedbuffer) {

buffer := passedbuffer //assigning passedbuffer to //buffer to constructor

}

// Create producer thread

Function run() { //overriding function run

Buffer produceData() // calling function produceData

}

}

//3) class

class Consumer {

buffer //create an object of buffer

Consumer (Buffer passedbuffer) {

buffer := passedbuffer //assigning passedbuffer to //buffer to constructor

}

// Create producer thread

Function run() { //overriding function run

Buffer consumeData() // calling function consumeData

}

}

}

//4) class

class main {

public static void main(String[] args) {

// Object of a class that has both produce()

// and consume() method

buffer //creating new object of buffer

// creating 4 threads 2 acts as producer and 2 acts as conumer

// sharing same buffer that has declared previous

Thread t1 := Thread(Producer(buffer), "producer 1")

Thread t2 := Thread(Producer(buffer), "producer 2")

Thread t3 := Thread(Consumer(buffer), "consumer 1")

Thread t4 := Thread(Consumer(buffer), "consumer 2")

// Start threads

t1 start()

t2 start()

t3.start()

t4.start()

}

}

Examples of Deadlock:

The possibility of deadlock in the producer consumer problem depends on what kind of implementation you are looking at and how you have used it.

Let us assume that we have one producer and one consumer with a single shared buffer between them.

Using wait() and notify() implementation:

The deadlock might happen if there is no consistency between the producer and consumer leading them to access the critical section simultaneously. This will lead to a scenario where both the consumer and the producer go to sleep and hence leading to deadlock.

Solution:

Using synchronized java method

It is used to lock an object for shared buffer, The buffer gets locked when the synchronized method is called, lock won’t be released until the thread completes its function.

Examples of starvation:

1.Consumer on certain condition is not returning the previously consumed buffer to empty buffer queue and continuing to wait for next ready buffer ready to be consumed.

2.Or Producer on certain condition is not returning produced buffer to ready buffer queue and continuing to wait for empty buffer to produce.

Then eventually this kind of situation will lead to starvation.

This kind of "waiting while holding buffer" scenario can lead to starvation.

Solution:

Consumer blocking till there is ready buffer to consume.

Producer blocking till there is empty buffer to produce.

Producer after producing immediately moving buffer to ready buffer queue.

Consumer after consuming immediately moving buffer to empty buffer queue.

This arrangement cannot lead to starvation.

Explanation for real world application:

Assume that we have a store people get in to buy some kind of products doesn’t matter which it is assume it’s clothesO, and store has stock that is limited at the same time the customers are buying the factory delivers products till the stock is full and customers are buying till the stock is empty or for period of time

Now let’s clear out each role for the probem the stock is the bounded buffer, the factory is the producer, and the customer is the consumer.