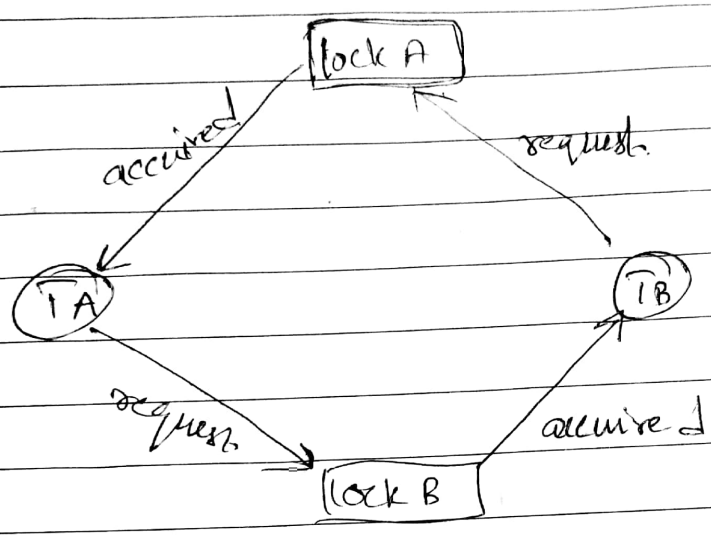
Uzair Ahmed Mughal

18K-0179

Section-4B.

OS - Mock Test

Q1(a)

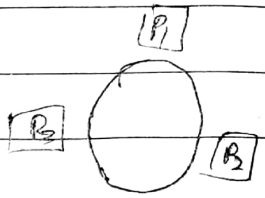


according to code Thread A acquires lock A first and Thread B acquires lock B, and when Thread A tries to acquire lock B Deadlock occurs as thread B is also trying to acquire Lock A.

(b) For this problem the solution will be to use one lock and when one Thread is done with it's critical region other thread will execute.

Thread A	Thread B
accuire lockA	accuire lockA
a=50	a=50
b=60	b=60
release lockA	releas lockA.

Q2. chopstick [3] = {1, 1}.



repeat {

wait(chopstick [i]). // accuire (set to 0).
wait (chopstick [(i+1) mod 3])

---

eat

---

signal (chopstick[i]) // release (set to 1)
signal ((chopstick [(i+1) mod 3])

---

think

---

until false}.

Note: This will be the structure of all person.

Signature

Date

Q3. 256 Pages, 18k-0179 = 128 kB
Physical memory = 128 frames.

(a) Logical address bits:

L.A Space = $256 = 2^8$
Page size = 128 kB = $2^7 \times 2^{10} = 2^{17}$
Bits = 17+8 = 25 Bits.

(b) Physical address bits:

P.M Space = $128 = 2^7$
Page size = 128 kB = $2^7 \times 2^{10} = 2^{17}$
Bits = 7+17 = 24 Bits

Q4:

Ans: First we need to understand what is internal fragmentation, when requested memory is smaller than allocated memory and the remaining memory will not be used that causes internal fragmentation. The best allocation method used by Os is paging which has far less internal fragmentation than Segmentation. In paging every page has a fixed size.

Physical memory is divided into fixed-size blocks called frames.

RC

Signature

Date

Logical memory is divided into blocks of same size called pages.

⇒ for example lets see how is internal fragmentation reduced.

Page size = 2048 bytes.
Process size = 74800 bytes

Number of frames = Process size / Page size = 74800 / 2048 = 36 + 1072 Bytes

Frames = 36 + 1072 Bytes (1 frame).

Internal fragmentation : 2048 - 1072 = 976 bytes.

Here we see it's an average case fragmentation

(b) Ans: Lets suppose your PC has multicore processor and has 8 Threads. If we process without threads we are not fully using our processor's capability. But if we run a multi treaded process we can use those threads provided by our processor. Threads are faster and cheaper in terms of resources. More utilization of multipe cores for parallel execution That increases the execution speed of programs If one thread gets blocked other threads will execute properly where as process alone can be blocked completely.

RC

Signature