Readers Writers Problem

1-Pseudocode :

1. process requests the entry to critical section.
2. If allowed i.e. wait() gives a true value, it enters and performs the write or read. If not allowed, it keeps on waiting.
3. It exits the critical section.

do {

// process requests for critical section

wait(wrt);

// performs the task

// leaves the critical section

signal(wrt);

} while(true);

-----------------------------------------------------------

2-Deadlock :

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Here is an example

1. **public** **class** TestDeadlockExample1 {
2. **public** **static** **void** main(String[] args) {
3. **final** String resource1 = "ratan jaiswal";
4. **final** String resource2 = "vimal jaiswal";
5. // t1 tries to lock resource1 then resource2
6. Thread t1 = **new** Thread() {
7. **public** **void** run() {
8. **synchronized** (resource1) {
9. System.out.println("Thread 1: locked resource 1");
11. **try** { Thread.sleep(100);} **catch** (Exception e) {}
13. **synchronized** (resource2) {
14. System.out.println("Thread 1: locked resource 2");
15. }
16. }
17. }
18. };
20. // t2 tries to lock resource2 then resource1
21. Thread t2 = **new** Thread() {
22. **public** **void** run() {
23. **synchronized** (resource2) {
24. System.out.println("Thread 2: locked resource 2");
26. **try** { Thread.sleep(100);} **catch** (Exception e) {}
28. **synchronized** (resource1) {
29. System.out.println("Thread 2: locked resource 1");
30. }
31. }
32. }
33. };

36. t1.start();
37. t2.start();
38. }
39. }

3-In my project I allowed only one resource shared between processes so when p1 lock this resource

And p2 want this resource so p1 don’t depend on p2

-------------------------------------------------------------------------------

4- Examples of Starvation :

What is Starvation : it’s when process is prevented to enter it’s critical section

Here is an example

**Writer process:**

1. Writer requests the entry to critical section.
2. If allowed i.e. wait() gives a true value, it enters and performs the write. If not allowed, it keeps on waiting.
3. It exits the critical section.

do {

// writer requests for critical section

wait(wrt);

// performs the write

// leaves the critical section

signal(wrt);

} while(true);

**Reader process:**

1. Reader requests the entry to critical section.
2. If allowed:
   * it increments the count of number of readers inside the critical section. If this reader is the first reader entering, it locks the **wrt** semaphore to restrict the entry of writers if any reader is inside.
   * It then, signals mutex as any other reader is allowed to enter while others are already reading.
   * After performing reading, it exits the critical section. When exiting, it checks if no more reader is inside, it signals the semaphore “wrt” as now, writer can enter the critical section.
3. If not allowed, it keeps on waiting.

do {

// Reader wants to enter the critical section

wait(mutex);

// The number of readers has now increased by 1

readcnt++;

// there is at least one reader in the critical section

**// this ensure no writer can enter if there is even one reader**

**// thus we give preference to readers here**

if (readcnt==1)

wait(wrt);

// other readers can enter while this current reader is inside

// the critical section

signal(mutex);

// current reader performs reading here

wait(mutex);   // a reader wants to leave

readcnt--;

// that is, no reader is left in the critical section,

if (readcnt == 0)

signal(wrt);         // writers can enter

signal(mutex); // reader leaves

if there is too many readers then writer process will have to wait for all readers to finish and they may be not finish

5-How I Solved it ?

By make each thread enter critical section alone

1. Writer or Reader requests the entry to critical section.
2. If allowed i.e. wait() gives a true value, it enters and performs the process . If not allowed, it keeps on waiting.
3. It exits the critical section.

do {

// writer requests for critical section

wait(wrt);

// performs the process

// leaves the critical section

signal(wrt);

} while(true);

------------------------------------------------------------