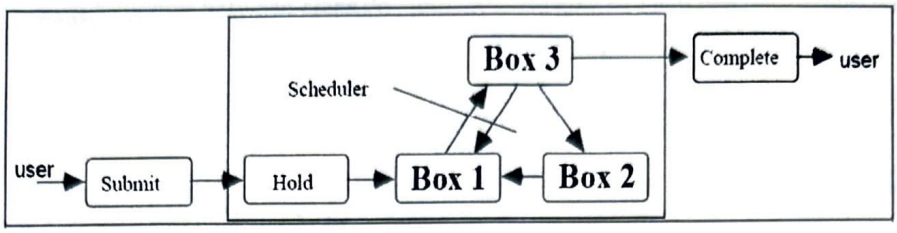
19

01741589 97

Name: Rajiv Karthik Reddy Kodimala **Midterm 2**



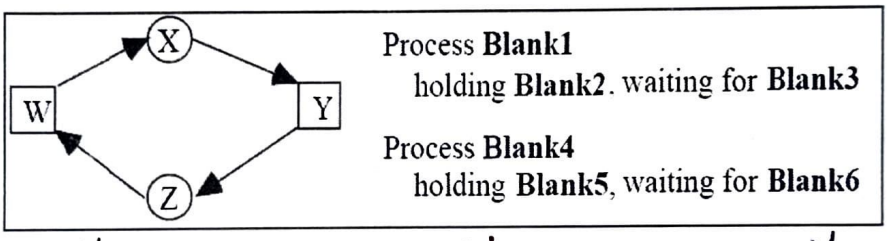
1) The diagram above shows three boxes labeled, one for each state that a process can be in while "active" in the computer system. We used this diagram many times throughout the semester to understand several topics such as Hyper-Threading, Deadlocks, Race Conditions and I/O Buffering. List the process state that each box represents, and the maximum number of processes (1 or more than 1) that can be in each state assuming a single-core, single-CPU system. [6 pts]

Box 1: Ready Number of processes: more than 1

Box 2: Wait Number of processes: more than 1

Box 3: Running Number of processes: 1

2) The diagram below shows six blanks in the context of a Holt Resource Graph for a set of processes and resources in a computer system. Describe the current state of the system by filling in the following six blanks with the appropriate letter: W, X, Y, or Z. Note that some letters will be used for more than one blank, and that there is more than one answer possible. [6 pts]



Blank 1: X Blank 2: W Blank 3: Y

Blank 4: Z Blank 5: Y Blank 6: W

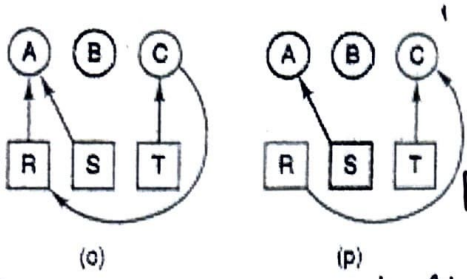
3a) Race Conditions and Deadlocks are non-deterministic bugs. Why are they non-deterministic as seen by the programmer and/or end-user? That is, what makes them non-deterministic? [4 pts]

Operating System handles the thread and random Inter process communication. Due to this from programmer perspective we cannot identify the sequence of operations in program which makes them seem as non-deterministic bugs to programmer

b) Explain why Race Conditions are considered a harder and more complex bug than Deadlocks. [4 pts]

Because of the nature of race conditions it results in incorrect outputs. It is hard to pinpoint which caused the output to change as these are very subtle. Hence race conditions are considered harder and more complex bugs than deadlock

4) The diagram below shows a state change in the context of a Holt Resource Graph for a set of processes and resources in a computer system between time steps "o" and "p". Explain <u>which process</u> executed <u>what specific action</u> between steps "o" and "p". Assuming each process shown will only need a maximum of two resources, <u>which process can now run</u> at step "p", but could not at step "o" ? [5 pts]



Between steps 'o' and 'p' Process A released Resource 'R' and process C acquired resource 'R'
Process A executed.

Process C can run at Step 'p' as it has both resources 'T' and 'R'
Process A can run at step 'p' as it has required resource 'S'
Process 'A' can run at step 'o' as it has both resources 'R' and 'S'
Process 'T' cannot run at step 'o' as it has only 'T' but not 'R'

5a) Write the pseudo-code (as shown in class) for the semaphore operation performed by P(S). [3 pts]

S: S-1
if (P(S) < 0)
process is in critical region and puts it in wait state

b) Write the pseudo-code (as shown in class) for the semaphore operation performed by V(S). [3 pts]

S: S+1
if (P(S) ≤ 0) signals the process in wating queue and puts it to run.

6a) The semaphore operations P and V are "special" functions. In particular, there is one very important property of semaphores that will guarantee mutual exclusivity between processes executing a P function, between processes executing a V function, and between processes executing a P function and a V function. What is this important property? You may reference the pseudo-code that you wrote in the previous problem and explain this property in the context of the pseudocode. [5 pts]

The special semaphore operations P and V are atomic in nature. We can use the MUTEX Semaphore to guarantee mutual exclusivity between processes. The operation 'P' ensures that till variable gt 0 it is running in critical section else it puts in wating queue. Operation 'V' once condition is satisfied it runs process bringing it from waitting queue.

b) How does the operating system implement or achieve the important property mentioned above ? [2 pts]

Operating System maintains a global Variable between processes to ensure mutual exclusivity between process executing P and between process executing a V function.

7) Shown below is the semaphore solution to the Readers/Writer problem as discussed in class.

Reader:

```
P(MUTEX)                          ⟶ Reader 2
   RC = RC + 1
   IF (RC = 1) THEN P(DB)         ⟶ Reader1
V(MUTEX)
   <read_data_base>
P(MUTEX)
   RC = RC - 1
   IF (RC = 0) THEN V(DB)
V (MUTEX)
GOTO Reader
```

Writer:

```
P(DB)
   <write_data_base>
V(DB)
GOTO Writer
```

a) What does the RC variable represent? [2 pts]

RC variable represents the readers count and increments everytime Reader is called and decrements accordingly.

b) What could potentially happen if the MUTEX semaphore were not utilized? [3 pts]

Mutual Exclusion MUTEX Semaphore ensures that one is resource is acquired by only 1 process at given time. If we do not use MUTEX then it would violate mutual exclusion property.

c) Consider a situation where a Writer is currently in the process of writing. Two readers, Reader1 followed by Reader2, now arrive in rapid succession. Draw an arrow(s) to the specific P function(s) where the two readers would be blocked in the code above. Label the arrow(s) with Reader1 and Reader2. Explain why each Reader gets blocked at that point. [6 pts]

Reader 1 gets blocked at statement if (RC=1) Then P(DB) as at this Condition statement Reader 1 is waiting for V(DB) and is blocked Reader 2 gets blocked at P(MUTEX) as it is waiting for the V(MUTEX) to be called. and waiting to be blocked.

d) The Readers-Writers solution as shown above has a potential flaw. What is the flaw and under what circumstances could it arise? [5 pts]

If in case there are infinite readers, writer would never get a chance to write to data base and hence as a result it waits indefinitely.

8a) Shown below is the semaphore solution to the Producer/Consumer problem as discussed in class. Three semaphores are used: EMPTY, FULL and MUTEX. Fill in the eight blank lines (denoting Semaphore Call 1 - 8) with the P and V calls to EMPTY, FULL and MUTEX. [8 pts]

Producer:	Consumer:
<non-critical>	<non-critical>
Semaphore Call 1	Semaphore Call 5
Semaphore Call 2	Semaphore Call 6
<Fill Buffer>	<Consume Buffer>
Semaphore Call 3	Semaphore Call 7
Semaphore Call 4	Semaphore Call 8
<non-critical>	<non-critical>
GOTO Producer	GOTO Consumer

Semaphore Call 1: P(EMPTY)
Semaphore Call 2: P(MUTEX)
Semaphore Call 3: V(MUTEX)
Semaphore Call 4: V(FULL)

Semaphore Call 5: P(FULL)
Semaphore Call 6: P(MUTEX)
Semaphore Call 7: V(MUTEX)
Semaphore Call 8: V(EMPTY)

b) Assume the capacity of the Buffer (in both the *<Fill Buffer>* and *<Consume Buffer>*) is four. Provide the initial values for the following Semaphores. [3 pts]

-3

EMPTY: 34 FULL: 50 MUTEX: 41

c) In the context of how the Buffer Data Structure would be implemented in computer code, what worst-case or boundary-case stress test is being applied to the Buffer if:

The Producer Process runs faster than the Consumer Process [2 pts]

(gt or equal)

This indicates that P(EMPTY) and P(MUTEX) are both ≥0 making sure Producer is not transferred to waiting state.

The Consumer Process runs faster than the Producer Process [2 pts]

Similarly P(FULL) and P(MUTEX) are both ≥0 making sure they are not moved to waiting state

9) What is a critical section ? [6 pts]

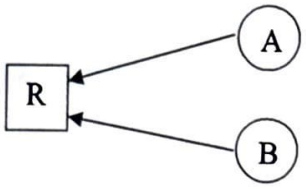
Critical Sections represents the frame where a process requests a resource and releases it. Critical Section ensures that one instance of resource is shared by one ~~process~~ using shared variables ensuring mutual exclusivity. In holt-resource graph if the path crosses this Critical Section it results in deadlock.

10) Four conditions are necessary for deadlock. In class, an example of two people trying to cross a river by stepping on stones was presented. Deadlock occurs if two people cross from opposite directions and meet somewhere in the middle. Which of the four conditions is illustrated by the constraint: [4 pts]

a) Only one person can step on a given stone at any one time: Mutual exclusion

b) One person cannot push the other into the water to take their stones: Non preemption

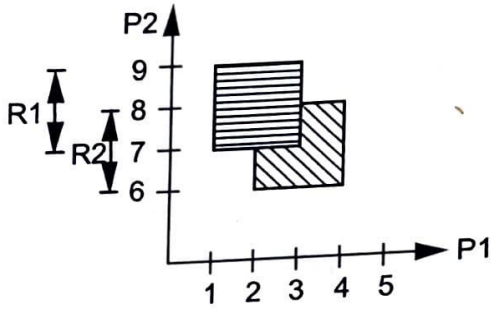
11) Is it possible for the following situation to arise in a Holt Resource Graph? Explain why and what it would represent, or why not and what condition it violates. [6 pts]



Yes following situation is possible to arise in Holt Graph. It represents that both process 'A' and 'B' are waiting for the resource 'R'.

As resource 'R' is currently not held by any Process mutual exclusion is not violated.

12) A resource trajectory plot is shown below for two processes, P1 and P2, which each need resources R1 and R2 at some point in their execution.



a) At what statement is resource R1 requested by P1 ? [2 pts] 1

b) At what statement is resource R2 released by P1 ? [2 pts] 4

c) What is the meaning and significance behind the two shaded regions in the diagram? [5 pts]

Two shaded regions represents the critical section of the Processes holding the resources. If trajectory crosses these shaded region or rectangle formed by 7,6,1,2 it results in deadlock.

13) Show (via pseudo-code) how two semaphores, if used incorrectly across two processes, could lead to deadlock. [6 pts]

Process1	Process2
P(S1)	P(S2)
P(S2)	P(S1)
V(S2)	V(S1)
V(S1)	V(S2)

Initially S1=0 then for Process1 P(S1) would make S1=-1 and process1 is in wait state

Similarly for Process 2 P(S2) would be S2=-1 and P2 is in wait state

Leading to deadlock