WAYGROUND formerly Quizizz **Worksheets**

Name

Class

Date

OS Unit-3

Total questions: 76

Worksheet time: 6hrs 20mins

1. Disabling interrupts frequently can affect the system's clock because:

   a) It prevents the clock interrupt from being serviced, causing the system clock to lose time.

   b) It speeds up the system clock by allowing more interrupts to occur.

   c) It has no effect on the system clock as interrupts are unrelated to timekeeping.

   d) It causes the system clock to reset to zero each time interrupts are disabled.

2. Busy waiting refers to a process repeatedly checking for a condition without relinquishing the CPU. What other kinds of waiting exist in an operating system, and can busy waiting be avoided altogether?

   a) Busy waiting can be avoided by using blocking or sleeping mechanisms, which allow the CPU to be used by other processes.

   b) Busy waiting is the only form of waiting in operating systems and cannot be avoided.

   c) Busy waiting can be avoided only by shutting down the operating system.

   d) Busy waiting can be avoided by increasing the CPU speed.

3. Spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems because:

   a) On single-processor systems, a process holding a spinlock cannot be preempted, causing deadlock, while multiprocessor systems allow other processors to run.

   b) Spinlocks are faster on single-processor systems than on multiprocessor systems.

   c) Multiprocessor systems do not require synchronization mechanisms like spinlocks.

   d) Spinlocks are only used for I/O operations in multiprocessor systems.

4. Show that, if the wait() and signal() semaphore operations are not executed atomically, then mutual exclusion may be violated.

   a) Mutual exclusion may be violated because concurrent execution can lead to race conditions.

   b) Mutual exclusion is always preserved regardless of atomicity.

   c) Atomicity of semaphore operations is not related to mutual exclusion.

   d) Mutual exclusion is only violated if there are more than two processes.

5. Illustrate how a binary semaphore can be used to implement mutual exclusion among n processes.

a) A binary semaphore allows only one process to enter the critical section at a time, ensuring mutual exclusion.
b) A binary semaphore allows all n processes to enter the critical section simultaneously.
c) A binary semaphore randomly selects processes to enter the critical section without ensuring mutual exclusion.
d) A binary semaphore is used only for process synchronization and cannot implement mutual exclusion.

6. A race condition is possible when two or more processes access shared resources simultaneously. What can be done to prevent race conditions from occurring?

a) Use synchronization mechanisms like locks or semaphores.
b) Increase the number of processes accessing the resource.
c) Ignore the shared resource and continue execution.
d) Allow unrestricted access to shared resources.

7. Windows and Linux implement multiple locking mechanisms. Which of the following best explains the circumstances under which spinlocks, mutex locks, semaphores, and condition variables are used, and why these mechanisms are needed?

a) Spinlocks are used for short critical sections where waiting is minimal; mutex locks are for longer waits; semaphores manage resource access; condition variables synchronize threads based on conditions.
b) Spinlocks are used only for file access; mutex locks are for network operations; semaphores are for memory allocation; condition variables are for hardware interrupts.
c) Spinlocks are used for user-level processes; mutex locks are for kernel-level processes; semaphores are for device drivers; condition variables are for GUI events.
d) Spinlocks, mutex locks, semaphores, and condition variables are all used interchangeably without specific circumstances or needs.

8. Providing a synchronization tool offers which of the following benefits?

a) Improved coordination among team members
b) Increased hardware requirements
c) Reduced communication between teams
d) Decreased productivity

9. Describe what changes would be necessary to the producer and consumer processes in Figure 7.1 and Figure 7.2 so that a mutex lock could be used instead of a binary semaphore.

a) The producer and consumer processes would need to use mutex lock operations (lock/unlock) instead of wait/signal operations for mutual exclusion.
b) The producer and consumer processes would need to remove all synchronization mechanisms.
c) The producer and consumer processes would need to use multiple mutex locks for each shared variable.
d) The producer and consumer processes would need to use only condition variables instead of any locking mechanism.

10. Deadlock is possible with the dining-philosophers problem when:

a) Each philosopher holds one fork and waits for the other, causing a circular wait.
b) Philosophers eat without waiting for forks.
c) Philosophers share all forks simultaneously.
d) Philosophers never pick up forks.

11. The difference between signaled and non-signaled states with Windows dispatcher objects is:

a) Signaled state allows threads to proceed, while non-signaled state causes threads to wait.
b) Signaled state causes threads to wait, while non-signaled state allows threads to proceed.
c) Both states allow threads to proceed without waiting.
d) Neither state affects thread execution.

12. Assume val is an atomic integer in a Linux system. What is the value of val after the following operations have been completed?

a) 7
b) 0
c) 1
d) -1

13. List three examples of deadlocks that are not related to a computer-system environment.

a) Traffic gridlock at an intersection, two people trying to pass each other in a narrow hallway, and two companies waiting for each other's approval to proceed.
b) Computer processes waiting for resources, printer queue deadlock, and database transaction deadlock.
c) Network packet collision, memory allocation failure, and CPU scheduling conflict.
d) File locking in operating systems, thread synchronization issues, and hardware resource contention.

14. Suppose that a system is in an unsafe state. It is possible for the threads to complete their execution without entering a deadlocked state.

a) True
b) False
c) Only if resources are released early
d) Only in single-threaded systems

15. A possible method for preventing deadlocks is to have a single, higher-order resource that must be requested before any other resource. For example, if multiple threads attempt to access the synchronization objects A ... E, deadlock is possible. (Such synchronization objects may include mutexes, semaphores, condition variables, and the like.) We can prevent deadlock by adding a sixth object F. Whenever a thread wants to acquire the synchronization lock for any object A ... E, it must first acquire the lock for object F. This solution is known as containment: the locks for objects A ... E are contained within the lock for object F. Compare this scheme with the circular-wait scheme of Section 8.5.4.

a) This is probably not a good solution because it yields too large a scope. It is better to define a locking policy with as narrow a scope as possible. The circular wait approach is a reasonable approach to avoiding deadlock, and does not increase the scope of holding a lock.

b) This is the best solution because it guarantees that no thread will ever be blocked.

c) This method is more efficient than circular-wait because it allows threads to acquire locks in any order.

d) Containment is always preferable to circular-wait because it simplifies the locking mechanism.

16. Prove that the safety algorithm requires an order of $\mathrm{m} \times \mathrm{n}^2$ operations.

a) The safety algorithm requires an order of m × n^2 operations.

b) The safety algorithm requires an order of m × n operations.

c) The safety algorithm requires an order of n × m^2 operations.

d) The safety algorithm requires an order of $\mathrm{m}^2 \times \mathrm{n}^2$ operations.

17. What are the arguments for installing the deadlock-avoidance algorithm?

a) It prevents system deadlocks by ensuring resources are allocated safely.

b) It increases the likelihood of deadlocks occurring in the system.

c) It reduces the efficiency of resource allocation without preventing deadlocks.

d) It allows processes to ignore resource allocation protocols.

18. Arguments against installing the deadlock-avoidance algorithm include:

a) It can be complex and costly to implement.

b) It guarantees system safety at all times.

c) It increases system performance significantly.

d) It eliminates all types of deadlocks automatically.

19. A system can detect that some of its threads are starving. How can it do this?

a) By monitoring thread progress and identifying threads that are not making progress over time.

b) By randomly terminating threads to prevent starvation.

c) By increasing the priority of all threads equally.

d) By ignoring thread starvation and focusing only on deadlock detection.

20. Deadlock can occur. Which of the following is a necessary condition for deadlock?

a) Mutual exclusion
b) No processes waiting
c) Unlimited resources
d) Processes do not hold resources

21. Indefinite blocking can occur. Explain your answer.

a) Yes, indefinite blocking can occur when processes are waiting for resources that may never become available.
b) No, indefinite blocking cannot occur because resources are always allocated in a timely manner.
c) Indefinite blocking only occurs in single-threaded systems, not in multi-threaded systems.
d) Indefinite blocking is impossible if all processes have equal priority.

22. Suppose that you have coded the deadlock-avoidance safety algorithm that determines if a system is in a safe state or not, and now have been asked to implement the deadlock-detection algorithm. Can you do so by simply using the safety algorithm code and redefining Max_i = Waiting_i + Allocation_i, where Waiting_i is a vector specifying the resources for which thread i is waiting and Allocation_i is as defined in Section 8.6?

a) No, because the safety algorithm checks for future safe states, while deadlock detection checks for current deadlocks, and redefining Max_i does not account for this difference.
b) Yes, because redefining Max_i as Waiting_i + Allocation_i allows the safety algorithm to detect deadlocks accurately.
c) No, because the safety algorithm cannot be used for deadlock detection under any circumstances.
d) Yes, because both algorithms fundamentally operate in the same way regardless of the definition of Max_i.

23. Is it possible to have a deadlock involving only one single-threaded process?

a) Yes, a single-threaded process can experience deadlock if it waits for a resource it already holds.
b) No, deadlock requires at least two processes or threads competing for resources.
c) Yes, deadlock can occur in any process regardless of the number of threads.
d) No, deadlock is only possible in multi-threaded processes.

24. What is the primary purpose of using semaphores in operating systems?

a) To increase the speed of CPU scheduling.
b) To handle input/output operations exclusively.
c) To provide a mechanism for process synchronization.
d) To manage memory allocation efficiently.

25. In the context of deadlocks, what does the term 'circular wait' refer to?

a) A strategy to allocate resources in a non-circular manner.
b) A situation where processes are waiting for resources in a circular chain.
c) A condition where all processes are executing without waiting.
d) A method to prevent deadlocks by allowing processes to wait indefinitely.

26. Which of the following strategies can be employed to resolve a deadlock once it has been detected?

a) Forcefully preempt resources from one or more processes.
b) Terminate all processes involved in the deadlock.
c) Allow processes to continue executing until they complete.
d) Increase the number of resources available to all processes.

27. What is the primary difference between deadlock prevention and deadlock avoidance?

a) Deadlock prevention and avoidance are the same and can be used interchangeably.
b) Deadlock prevention is only applicable in single-threaded environments, while deadlock avoidance is for multi-threaded environments.
c) Deadlock prevention allows processes to wait indefinitely, while deadlock avoidance does not.
d) Deadlock prevention ensures that at least one of the necessary conditions for deadlock cannot hold, while deadlock avoidance dynamically checks resource allocation to ensure a safe state.

28. In a system with multiple processes, which of the following scenarios can lead to a deadlock?

a) Process A does not request any resources.
b) Process A completes its execution and releases all resources.
c) Process A holds Resource 1 and waits for Resource 2, while Process B holds Resource 2 and waits for Resource 1.
d) Process A and Process B both request the same resource simultaneously.

29. Which of the following statements about semaphores is true?

a) Semaphores are less efficient than mutexes in all scenarios.
b) Semaphores can be used to implement mutual exclusion and can also be used for signaling between processes.
c) Semaphores are only useful for signaling and cannot implement mutual exclusion.
d) Semaphores are always binary and cannot have a value greater than one.

30. What is the Dining Philosopher Problem in computer science?

a) A challenge to create a new type of dining table for computer science labs
b) A problem related to finding the best dining options for programmers
c) A cooking competition among computer scientists
d) A classic synchronization problem in computer science

31. Which statement best defines a deadlock in concurrent processing?

a) Processes communicate freely without any synchronization
b) Processes wait cyclically for resources held by each other
c) Processes run sequentially without any shared resources
d) Processes terminate immediately after receiving all resources

32. Which topics are typically included when studying deadlocks in operating systems?

a) System model and resource allocation
b) Deadlock characterization and detection
c) Compiler optimization strategies
d) Prevention, avoidance, and recovery methods

33. What is a primary goal when learning about deadlocks in a computer system?

a) Reduce the number of user applications installed
b) Eliminate all process synchronization primitives
c) Increase CPU clock speed for faster execution
d) Develop methods to prevent or avoid deadlocks

34. Which sequence correctly describes how a process uses a resource in the system model?

a) request, use, release
b) acquire, hold, preempt
c) wait, execute, unlock
d) allocate, compute, free

35. Which set lists valid resource types in the system model?

a) CPU cycles, memory space, I/O devices
b) threads, frames, semaphores
c) files, sockets, messages
d) pages, pipes, interrupts

36. In mutual exclusion, what is enforced for each resource instance?

a) Many processes can share it
b) Only one process can use it
c) Preemption is always allowed
d) It must be released immediately

37. Which statement best describes hold and wait?

a) Holding at least one resource while waiting for more
b) Holding no resources while waiting in a queue
c) Preempting other processes to acquire resources
d) Releasing all resources before requesting new ones

38. Which condition prohibits the system from forcibly taking a resource from a process?

a) Hold and wait
b) Circular wait
c) Mutual exclusion
d) No preemption

39. Which pattern defines circular wait among processes P0, P1, …, Pn?

a) Each waits for a resource held by the next
b) All wait for the same single resource
c) Each releases resources in reverse order
d) None can request until all have finished

40. Select all conditions that must hold simultaneously for a deadlock to occur.

a) Mutual exclusion
b) Hold and wait
c) No preemption
d) Circular wait

41. Deadlocks can arise with mutex locks primarily due to what mechanism?

a) Network packet scheduling
b) Compiler optimization passes
c) User interface redraw events
d) System calls and locking behavior

42. In a resource-allocation graph, what are the two types of vertices represented?

a) Processes and threads
b) Requests and assignments
c) Processes and resources
d) Resources and instances

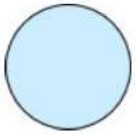
43. Which directed edge represents a process requesting a resource instance?

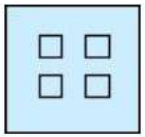
a) Pi → Pk arrow
b) Pi ↔ Rj bidirectional
c) Rj → Pi arrow
d) Pi → Rj arrow

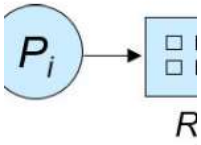
44. Which directed edge indicates a resource instance currently allocated to a process?

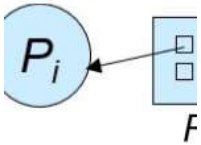
a) Rj → Rk arrow
b) Pi → Rj arrow
c) Rj → Pi arrow
d) Pi ↔ Rj bidirectional

45. A resource type with four instances is shown as a square with four small squares inside. What does each small square represent?







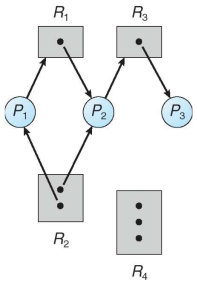


a) An assignment edge

b) A process token

c) A resource instance

d) A request edge

46. 

In the example graph, arrows from P1 and P2 pointing to R1 indicate what relationship?

a) Resources requesting processes
b) Processes requesting P3
c) Processes holding R1
d) Processes requesting R1

47. Select all statements that correctly describe components of a resource-allocation graph.

a) Request edges point Pi → Rj
b) Resource types are rectangular vertices
c) Processes are circular vertices
d) Assignment edges point Rj → Pi

48. Which option best distinguishes a process Pi requesting Rj versus holding Rj in the notation used?

a) Request uses Pi → Rj; hold uses Rj → Pi
b) Request uses Rj → Pi; hold uses Pi → Rj
c) Both use undirected edges between Pi and Rj
d) Both use Pi → Rj arrows

49. In a resource-allocation graph, what condition guarantees there is no deadlock?

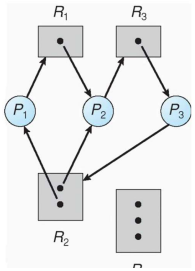
a) The graph has at most one cycle
b) Every process requests exactly one resource
c) The graph has no cycles anywhere
d) Every resource has multiple instances

50. Consider a cycle in a resource-allocation graph where each resource type has exactly one instance. What must be true?

a) The cycle implies a deadlock
b) The cycle implies safe allocation
c) The cycle implies starvation only
d) The cycle implies immediate completion

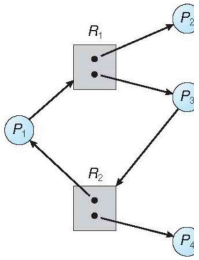
51. A cycle exists in a resource-allocation graph, and several instances exist for at least one resource type. Which statement is correct?

a) All processes can finish without waiting
b) Deadlock is impossible in any case
c) Deadlock must occur immediately
d) Deadlock might occur but is not guaranteed

52. 

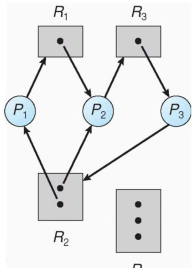
In the first diagram, P1, P2, and P3 form a cycle involving resources R1 and R3 with single instances. What conclusion is most consistent with the diagram?

a) A deadlock exists among the processes
b) Only P3 is blocked temporarily
c) R4 breaks the cycle automatically
d) The system is in a safe state

53. 

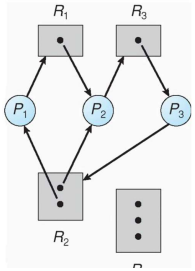
In the second diagram, there is a cycle but no deadlock. What feature of the resources explains this outcome?

a) Processes hold no resources
b) Edges are request-only, not assignment
c) All resources are unavailable
d) Multiple instances available in resources

54. 

Which change can prevent deadlock in a cycle among processes P1, P2, P3?

a) Increase instances of one resource type
b) Add a new process requesting nothing
c) Release a held resource by a process
d) Allow preemption of resources

55. 

Which statement correctly differentiates the two diagrams?

a) First guarantees deadlock given the cycle
b) Second guarantees safety despite the cycle
c) Second shows multi-instance resources allowing progress
d) First shows single-instance resources forming a cycle

56. Which is a valid high-level method for dealing with deadlocks in operating systems?

a) Allocate all resources to every process always
b) Guarantee deadlocks never occur via prevention
c) Randomly reassign processes without checks
d) Delay all I/O until program completion

57. Which approach allows the system to enter a deadlock and then fix it?

a) Deadlock recovery after detection
b) Deadlock avoidance with graphs
c) Circular wait enforcement
d) Mutual exclusion relaxation

58. Some systems ignore deadlocks entirely. Which statement best describes this policy?

a) Require strict resource ordering
b) Always preempt locks when a request arrives
c) Terminate any process holding resources
d) Assume deadlocks never occur in practice

59. Which statement about mutual exclusion in prevention is correct?

a) All resources must be mutually exclusive always
b) Mutual exclusion applies only to CPU time
c) Sharable resources need not be mutually exclusive
d) Files are never sharable under prevention

60. In hold-and-wait prevention, what must be guaranteed when a process requests a resource?

a) It holds a read-only file
b) It holds the CPU exclusively
c) It holds at least one other resource
d) It holds no other resources at that time

61. Which is a consequence of requiring processes to request all resources upfront to prevent hold-and-wait?

a) Low resource utilization, possible starvation
b) Elimination of mutual exclusion needs
c) Higher throughput for interactive jobs
d) Guaranteed maximal parallelism

62. Under no-preemption prevention, what happens if a process holding resources requests another unavailable resource?

a) All other processes are paused
b) The scheduler increases the process priority
c) The new request is permanently denied
d) Currently held resources are released

63. Which strategy prevents circular wait in deadlock prevention?

a) Always request all resources simultaneously
b) Allocate resources randomly to processes
c) Disallow sharable resources entirely
d) Impose total ordering of resource types

64. Two threads each use two mutexes named first_mutex and second_mutex. Thread one locks first_mutex then second_mutex; thread two locks second_mutex then first_mutex. What condition can occur when both threads run concurrently?

a) Starvation due to priority inversion
b) Livelock continuously retrying operations
c) Race condition updating shared data
d) Deadlock waiting on opposite locks

65. In the deadlock example, which ordering of lock acquisition prevents the deadlock between the two threads?

a) Alternating lock order between iterations
b) Always acquiring first_mutex before second_mutex
c) Acquiring whichever lock is currently free
d) Randomly choosing the first lock each time

66. Consider a transaction function that acquires lock1 for the 'from' account, then acquires lock2 for the 'to' account, performs withdraw and deposit, and releases lock2 then lock1. Which practice makes these concurrent transfers safe from deadlock?

a) Adding delays between lock acquisitions
b) Using non-blocking atomic operations
c) Ordering locks consistently across transactions
d) Holding both locks for the shortest time

67. Which statements are true about mutex locks in concurrent transactions transferring money between two accounts?

a) Lock ordering avoids circular wait conditions
b) Releasing locks in reverse acquisition order helps safety
c) Each transaction must acquire both account locks
d) Only one account lock is necessary for correctness

68. In the two-thread C example, what exact sequence avoids deadlock if both threads follow it?

a) Lock first, unlock first, lock second, unlock second
b) Lock first, lock second, unlock second, unlock first
c) Lock whichever is free, then lock the other later
d) Lock second, lock first, unlock first, unlock second

69. Which information must each process provide in deadlock avoidance to help the system plan resource allocation?

a) Minimum guaranteed resources for every process
b) Typical average resource usage per task
c) Exact execution time of each process
d) Maximum number of resources of each type

70. What does a deadlock-avoidance algorithm dynamically examine to prevent circular-wait?

a) Process queue order at runtime
b) Resource-allocation state over time
c) CPU scheduling priority levels
d) I/O device latency patterns

71. Which components define the resource-allocation state in deadlock avoidance?

a) Number of available resources
b) Maximum demands of processes
c) Order of process arrivals
d) Number of allocated resources

72. A system is in a safe state if there exists a sequence of all processes such that for each process Pi, which condition holds?

a) Pi must release its current resources before requesting more
b) Pi waits indefinitely until all other processes terminate
c) Pi can immediately obtain all requested resources
d) Pi's remaining request can be satisfied by available plus released resources

73. If Pi's needed resources are not immediately available in a safe state, what is the system's behavior?

a) Pi preempts resources from Pj
b) Pi switches to a different algorithm
c) Pi waits until some Pj finish
d) Pi aborts and restarts later

74. When a process Pj finishes in a safe sequence, what happens next?

a) The system enters an unsafe state
b) Pj retains its resources for reuse
c) Pi obtains needed resources and executes
d) All processes pause for synchronization

75. Which statement best captures the relationship between safe and unsafe states?

a) Safe state ensures maximum throughput
b) Safe state implies no deadlocks
c) Unsafe state guarantees deadlock
d) Unsafe state eliminates circular wait

76. What is the goal of deadlock avoidance with respect to system states?

a) Allow occasional unsafe states for efficiency
b) Ensure the system never enters an unsafe state
c) Convert unsafe states into deadlocks
d) Ensure entry into an unsafe state

Answer Keys

1. a) It prevents the clock interrupt from being serviced, causing the system clock to lose time.
2. a) Busy waiting can be avoided by using blocking or sleeping mechanisms, which allow the CPU to be used by other processes.
3. a) On single-processor systems, a process holding a spinlock cannot be preempted, causing deadlock, while multiprocessor systems allow other processors to run.
4. a) Mutual exclusion may be violated because concurrent execution can lead to race conditions.
5. a) A binary semaphore allows only one process to enter the critical section at a time, ensuring mutual exclusion.
6. a) Use synchronization mechanisms like locks or semaphores.
7. a) Spinlocks are used for short critical sections where waiting is minimal; mutex locks are for longer waits; semaphores manage resource access; condition variables synchronize threads based on conditions.
8. a) Improved coordination among team members
9. a) The producer and consumer processes would need to use mutex lock operations (lock/unlock) instead of wait/signal operations for mutual exclusion.
10. a) Each philosopher holds one fork and waits for the other, causing a circular wait.
11. a) Signaled state allows threads to proceed, while non-signaled state causes threads to wait.
12. a) 7
13. a) Traffic gridlock at an intersection, two people trying to pass each other in a narrow hallway, and two companies waiting for each other's approval to proceed.
14. a) True
15. a) This is probably not a good solution because it yields too large a scope. It is better to define a locking policy with as narrow a scope as possible. The circular wait approach is a reasonable approach to avoiding deadlock, and does not increase the scope of holding a lock.
16. a) The safety algorithm requires an order of m × n^2 operations.
17. a) It prevents system deadlocks by ensuring resources are allocated safely.
18. a) It can be complex and costly to implement.
19. a) By monitoring thread progress and identifying threads that are not making progress over time.
20. a) Mutual exclusion
21. a) Yes, indefinite blocking can occur when processes are waiting for resources that may never become available.
22. a) No, because the safety algorithm checks for future safe states, while deadlock detection checks for current deadlocks, and redefining Max_i does not account for this difference.
23. b) No, deadlock requires at least two processes or threads competing for resources.
24. c) To provide a mechanism for process synchronization.
25. b) A situation where processes are waiting for resources in a circular chain.
26. a) Forcefully preempt resources from one or more processes.
27. d) Deadlock prevention ensures that at least one of the necessary conditions for deadlock cannot

hold, while deadlock avoidance dynamically checks resource allocation to ensure a safe state.

28. c) Process A holds Resource 1 and waits for Resource 2, while Process B holds Resource 2 and waits for Resource 1.
29. b) Semaphores can be used to implement mutual exclusion and can also be used for signaling between processes.
30. d) A classic synchronization problem in computer science
31. b) Processes wait cyclically for resources held by each other
32. a) System model and resource allocation, b) Deadlock characterization and detection, d) Prevention, avoidance, and recovery methods
33. d) Develop methods to prevent or avoid deadlocks
34. a) request, use, release
35. a) CPU cycles, memory space, I/O devices
36. b) Only one process can use it
37. a) Holding at least one resource while waiting for more
38. d) No preemption
39. a) Each waits for a resource held by the next
40. a) Mutual exclusion, b) Hold and wait, c) No preemption, d) Circular wait
41. d) System calls and locking behavior
42. c) Processes and resources
43. d) Pi → Rj arrow
44. c) Rj → Pi arrow
45. c) A resource instance
46. d) Processes requesting R1
47. c) Processes are circular vertices, b) Resource types are rectangular vertices, a) Request edges point Pi → Rj, d) Assignment edges point Rj → Pi
48. a) Request uses Pi → Rj; hold uses Rj → Pi
49. c) The graph has no cycles anywhere
50. a) The cycle implies a deadlock
51. d) Deadlock might occur but is not guaranteed
52. a) A deadlock exists among the processes
53. d) Multiple instances available in resources
54. a) Increase instances of one resource type, c) Release a held resource by a process, d) Allow preemption of resources
55. d) First shows single-instance resources forming a cycle, c) Second shows multi-instance resources allowing progress, a) First guarantees deadlock given the cycle
56. b) Guarantee deadlocks never occur via prevention
57. a) Deadlock recovery after detection
58. d) Assume deadlocks never occur in practice
59. c) Sharable resources need not be mutually exclusive
60. d) It holds no other resources at that time

61. a) Low resource utilization, possible starvation

62. d) Currently held resources are released

63. d) Impose total ordering of resource types

64. d) Deadlock waiting on opposite locks

65. b) Always acquiring first_mutex before second_mutex

66. c) Ordering locks consistently across transactions

67. c) Each transaction must acquire both account locks , b) Releasing locks in reverse acquisition order helps safety , a) Lock ordering avoids circular wait conditions

68. b) Lock first, lock second, unlock second, unlock first

69. d) Maximum number of resources of each type

70. b) Resource-allocation state over time

71. a) Number of available resources , d) Number of allocated resources , b) Maximum demands of processes

72. d) Pi's remaining request can be satisfied by available plus released resources

73. c) Pi waits until some Pj finish

74. c) Pi obtains needed resources and executes

75. b) Safe state implies no deadlocks

76. b) Ensure the system never enters an unsafe state