L2910-5335 Exam 2 VerA

Phong Vo

TOTAL POINTS

75 / 100

QUESTION 1

1 Race condition 5 / 5

✓ - 0 pts Correct

QUESTION 2

2 Entry section 5 / 5

✓ - 0 pts Correct

QUESTION 3

3 Peterson's solution 0 / 5

✓ - 5 pts Incorrect (correct answer is "flag[j]==false || turn==i"

The entry section for Peterson's solution includes the following test in the busy-wait loop for the entry section for Process i:
while (flag[j] && turn == j) ;
Thus, by application of DeMorgan's Laws (https://en.wikipedia.org/wiki/De_Morgan%27s_laws), the busy-wait will end (and Process i will enter its critical section) when flag[j]==false || turn==i

QUESTION 4

4 Dining Philosophers 5 / 5

✓ - 0 pts Correct

QUESTION 5

5 Deadlock conditions - mutual exclusion 5 / 5

✓ - 0 pts Correct

QUESTION 6

6 Resource allocation graph 5 / 5

✓ - 0 pts Correct

QUESTION 7

7 GP OS strategy for deadlocks 5 / 5

✓ - 0 pts Correct

QUESTION 8

8 Safe-unsafe states 5 / 5

✓ - 0 pts Correct

QUESTION 9

9 Atomic class 2 / 2

✓ - 0 pts Correct

QUESTION 10

10 High level mutexes 2 / 2

✓ - 0 pts Correct

Although mutexes (as a software mechanism) are considered higher level than hardware mechanisms such as the test-and-set or compare-and-swap instructions, mutexes are lower level than more advanced software mechanisms such as futures.

QUESTION 11

11 Mutex = counting semaphore 2 / 2

✓ - 0 pts Correct

QUESTION 12

12 Hardware support for CS 2 / 2

✓ - 0 pts Correct

QUESTION 13

13 Counting semaphore value 2 / 2

✓ - 0 pts Correct

QUESTION 14

14 Total ordering for resources 0 / 2

✓ - 2 pts Incorrect (correct answer is True)

QUESTION 15

15 Unsafe state != deadlock 2 / 2

✓ - 0 pts Correct

QUESTION 16

16 Banker's algorithm & multiple resource instances 2 / 2

✓ - 0 pts Correct

QUESTION 17

17 Circular wait implies hold-and-wait 2 / 2

✓ - 0 pts Correct

QUESTION 18

18 wait-for graph not applicable to multiple resources 2 / 2

✓ - 0 pts Correct

QUESTION 19

19 Describe how Banker's algorithm avoids deadlocks 7 / 10

✓ - 3 pts Partially correct - see comments

QUESTION 20

Processor affinity 10 pts

20.1 (a) better utilization of per-processor cache 1 / 5

✓ - 4 pts Incorrect (see comments)

By scheduling a thread on the same processor on which it has previously run, the thread is able to take advantage of any memory accesses that have been cached on that processor during its previous run.

20.2 (b) common vs per-processor ready queues 1 / 5

✓ - 4 pts Incorrect (see comments)

Per-processor ready queues would provide better support, because the OS would automatically run a thread on the same processor each time it is ready.

QUESTION 21

21 SJF 10 / 10

✓ - 0 pts Correct (or close enough for full credit!)

SJF requires prior knowledge of CPU burst times for each process. Although feasible in batch systems, it is unreasonable to require the user(s) to provide this information in advance for modern interactive systems. (However, SJF may be approximated as "Shortest Remaining Time First" (SRTF) by use of exponential averaging of predicted burst times and actual past burst times.)
Even though it's not practical, SJF is still important because it gives the optimal performance, and thus serves as a benchmark or standard of comparison for other CPU scheduling algorithms.

QUESTION 22

22 Deadlock prevention vs avoidance 3 / 10

✓ - 7 pts Incorrect (see comments)

Both deadlock prevention and deadlock avoidance ensure that the system will never enter a deadlock state. Deadlock prevention does this by restraining requests such that one of the four necessary conditions for deadlock is removed (typically either hold-and-wait, no-preemption, or circular-wait). Deadlock avoidance does this by requiring all processes to provide a priori knowledge of their maximum resource needs, so that the avoidance algorithm can examine all resource requests for potential circular wait conditions.

Name: PHONG VO Student ID: 01790283 (a)

Exam #2

COMP.3080 – Operating Systems; November 15, 2019 – Dr. Wilkes

Note: *This exam is closed book and notes, except for one 8.5x11" sheet of paper with handwritten notes (no photocopies).*

Please write your name and student ID on each page of the exam!

Multiple Choice Questions – 5 points each.

1. **(MARK A SINGLE ANSWER)** A race condition _____.
   - ○ will result only if the outcome of execution does not depend on the order in which instructions are executed
   - ○ results when several threads try to access the same data concurrently
   - ● results when several threads try to access and modify the same data concurrently
   - ○ none of the above
2. **(MARK A SINGLE ANSWER)** A(n) ________ refers to code in which a process is **requesting** access to shared data.
   - ○ critical section
   - ● entry section
   - ○ mutex
   - ○ test-and-set
   - ○ none of the above
3. **(MARK A SINGLE ANSWER)** In Peterson's solution, process P_i (*P* sub `i`) may enter its critical section when _______.
   - ● `flag[j]==true && turn==j`
   - ○ `flag[j]==false || turn==i`
   - ○ `turn[i]==true && flag==i`
   - ○ `turn[j]==false || flag==j`
   - ○ none of the above
4. **(MARK A SINGLE ANSWER)** In the Dining Philosophers problem, what happens if each philosopher simultaneously picks up their right chopstick?
   - ○ Concurrency
   - ● Deadlock
   - ○ Mutual exclusion
   - ○ Preemption
   - ○ Priority inversion
   - ○ None of the above
5. **(MARK A SINGLE ANSWER)** Which of the following necessary conditions for deadlock states that at least one resource must be held in a non-shareable mode?
   - ○ Circular wait
   - ○ Hold and wait
   - ● Mutual exclusion
   - ○ No preemption

Name: PHONG VO Student ID: 01790283 (a)

6. **(MARK ALL THAT APPLY)** Which of the following is true in a system resource-allocation graph?
   - □ A directed edge from a process to a resource is called an assignment edge.
   - ■ A directed edge from a process to a resource is called a request edge.
   - ■ A directed edge from a resource to a process is called an assignment edge.
   - □ A directed edge from a resource to a process is called a request edge.
   - □ None of the above

7. **(MARK A SINGLE ANSWER)** Which of the following is the most common strategy used by general-purpose operating systems to handle deadlocks?
   - ○ Detect and recover from deadlocks
   - ● Pretend that deadlocks never occur
   - ○ Use protocols to prevent or avoid deadlocks
   - ○ None of the above

8. **(MARK A SINGLE ANSWER)** Which of the following statements is **true**?
   - ○ A safe state is a deadlocked state.
   - ○ A safe state may lead to a deadlocked state.
   - ○ An unsafe state is necessarily, and by definition, always a deadlocked state.
   - ● An unsafe state may lead to a deadlocked state.
   - ○ None of the above.

Name: PHONG VO Student ID: 017 90 283 (a)

True/False Questions – 2 points each.

9. The `atomic` class in the C++11 thread library is a version of the monitor construct.
   - ● True
   - ○ False

10. In the C++11 thread library, "high level" tools such as mutexes are less prone to programmer error than "low level" tools such as futures.
    - ○ True
    - ● False

11. Mutex locks and counting semaphores are basically equivalent.
    - ○ True
    - ● False

12. Practical solutions to the critical section problem require hardware support.
    - ● True
    - ○ False

13. The value of a counting semaphore can range only between 0 and 1.
    - ○ True
    - ● False

14. Ordering resources—and requiring the resources to be acquired in order—prevents the circular wait from occurring, and therefore prevents deadlock from occurring (assuming application programmers write programs that follow the ordering).
    - ○ True
    - ● False

15. A system in an unsafe state will ultimately deadlock.
    - ○ True
    - ● False

16. The banker's algorithm is useful in a system with multiple instances of each resource type.
    - ● True
    - ○ False

17. The circular-wait condition for a deadlock implies the hold-and-wait condition.
    - ● True
    - ○ False

18. The wait-for graph scheme is not applicable to a resource allocation system with multiple instances of each resource type.
    - ● True
    - ○ False

Name: ______________________________ Student ID: __________________________ (a)

Short Answer Questions – 10 points each: Write your answer in the space provided.

19. Briefly describe in words how the Banker's Algorithm avoids deadlocks in a system with multiple copies of resources. (You do not need to give details of the Banker's Algorithm or its data structures – just explain the general concepts.)

Because one of the reason causes a deadlock occur is the processes sharing the common resources. So, the Banker's algorithm's copying the resource will avoid the system from the deadlock.

Also, the Banker's algorithm is useful in a system with multiple instances of each resource type.

20. Consider the concept of *processor affinity* scheduling for threads (i.e., the OS attempts to schedule a thread on the same processor whenever possible).

a. Briefly describe how processor affinity scheduling enables better utilization of per-processor cache memory.

Processor affinity scheduling can adjust / impact to the process's priority levels.

b. Would a common ready queue, or per-processor ready queues, provide better support for processor affinity scheduling? Explain your answer.

- The computer hardware traps the kernel and PC is on stack. An assembly program
- starts to save the general registers and other volatile information (OS can not destroy it further)
- OS finds a page that has occured and tried to find out which virtual page needed
- Once virtual address causes page fault, system checks to see if the address is valid and no protection access problem. If valid, continues check to see if a page frame is free
  - no free: the page replacement algorithm is run
  - dirty: transfer to disk
  - clean: OS looks up disk address

Name: PHONG VO Student ID: 01790283 (a)

21. Briefly explain why the Shortest Job First (SJF) scheduling algorithm is not practical for modern interactive systems, but still is an important algorithm.

SJF is still important be cause the average time is least reduced rather than FCFS. The exponential everaging (lenth of the next CPU burst) is much reduced. SJF precede the shortest CPU burst time to be done first, the longest CPU burst in done inthelast time. This performance reduce the waiting time of every CPU burst, that may help overall CPU burst and tasks are finished a.s.a.p.

22. Briefly explain the difference between deadlock prevention and deadlock avoidance.

Deadlock prevention

- prevent a deadlock may occur by impacting the system's CPU burst and the priority levels of the process.
- Impacting to hold-and-wait condition.

Deadlock avoidance

- Avoid a deadlock may occurr by impacting the system's resources.