Sample Answers for New Sample Exam

Q1

1. False, the role of the operating system is to abstract away low level details
2. True (If the processor features a privileged mode, then the operation system will use it.)
3. False, the kernel has a separate stack
4. False, the operating system is language independent and has no control over the C library which is user code
5. True, otherwise syscalls wouldn't work
6. False, a process can go from blocked to ready but not the other way, it needs to be running to block.
7. True, in co-operative multitasking, a thread must yield() in order for another thread to take over
8. True, the kernel does not know about user-level threads
9. True, the OS can schedule kernel-level threads
10. False, user-level applications can not disable interrupts as this requires privileged mode
11. True, if initialised to one
12. False, condition variables are used in conjunction with *monitors*, or monitor-like lock use (e.g. in OS/161)
13. True, although this is not realistic in practice
14. False, in fact hold and wait is one of the four preconditions that leads to deadlock
15. True, metadata representing the empty blocks can be stored instead
16. True, even if the FS remains in a consistent (or recoverable) state, data is still lost
17. True, if the file increases in size, re-allocation will be required
18. False, Random access in a linked list is very slow so not 'desirable'
19. True, best fit results in the least external fragmentation.
20. False, swapping *swaps* the entire application, so it must fit entirely in memory to be present. Kevin: There is some confusion between swapping versus paging (note linux/windows use the term "swapping" inaccurately). The following link happens to have a nice answer https://stackoverflow.com/questions/4415254/difference-swapping-and-paging
21. True, the application controls the overlays
22. True, the OS checks that memory access falls within a segment's boundaries
23. False, they are two unrelated concepts
24. False, it is not practical
25. True, it doesn't have to be written to disk
26. True, the larger granularity of inclusion results in more virtual memory being included in the working set when only part of the page is actually accessed.
27. False, the TLB hit rate is higher if entries are reused before they are replaced
28. False, buffering adds memory copy overhead, which can be significant for fast I/O devices (e.g. multi gigabit networks).

29. False, Shortest Seek Time First can cause starvation
30. True, otherwise CPU time is wasted polling
31. False, a combination of I/O-bound and CPU-bound is optimal
32. False, but earliest-deadline-first does, see the example in lectures
33. True
34. False, they can avoid the overhead of context switching to scheduler then another process on a multiprocessor
35. True, ready processes are variable to all CPUs
36. False, reduces bus traffic since the test & set instructions are only applied when they might succeed.
37. True
38. False, the PT structure must be readable by the hardware refilling the TLB
39. True, the whole segment can be copied higher or lower in physical memory
40. True

Q2

(A) and (B): same as sample exam #1

(C)

Number of blocks addressable from one block = 2048 / 4 = 512

Direct blocks = 12

Single indirect blocks = 512

Double indirect blocks = 512**2

Triple indirect blocks = 512**3

Total blocks = 134480396

Max file size = 134480396 * 2KiB = 256.5GiB

Q3

Q4

(A) and (B): same as sample exam #1

(C) Two reasons:

#1. To decrease memory bus traffic. CPUs running processes competing for a simple test and set spinlock (TSL) repeatedly lock the bus (if test and set is not an atomic instruction on this hardware) or use the memory bus to synchronise their caches of the spinlock. Each CPU competing for a MCS lock keeps its *own* lock node in its local cache, hence there is no repeated cache synchronisation or main memory access.

#2. The make the lock fair (FIFO queue). A TSL is up for grabs when it is released, whereas a MSC lock is a queue.

Q5

(A): same as sample exam #1

(B)

#1 role: Providing an abstract machine consistent across differing hardware. Kevin: Note: System calls are not a *service*, they are a mechanism that provides access to OS services. A sample of a service that abstracts hardware is the file system. It provides an abstraction of named repositories of data instead of low level disk controllers, blocks, etc... The benefit is that applications are portable across differing hardware platforms and can work with a simpler, more programmer friendly, secure API.

#2 role: Manage the distribution of resources between processes. The service: the scheduler assigns CPU execution time to processes / threads. The benefit: a decent 'interactive' scheduler ensures the system appears responsive, parallelised and makes efficient use of the hardware by balancing execution and IO.

SampleAnswers2 (last edited 2017-06-23 21:07:08 by TedZhang)