# COMP3231/9201/3891/9283 Operating Systems 2020/T1

UNSW

## Tutorial Week 10

# Questions

### Virtual Memory

1. What effect does increasing the page size have?

---

2. Why is demand paging generally more prevalent than pre-paging?
3. Describe four replacement policies and compare them.

---

4. What is thrashing? How can it be detected? What can be done to combat it?

---

### I/O

5. Describe *programmed I/O* and *interrupt-driven I/O* in the case of receiving input (e.g. from a serial port). Which technique normally leads to more efficient use of the CPU? Describe a scenario where the alternative technique is more efficient.

---

6. A device driver routine (e.g. `read_block()` from disk) is invoked by the file system code. The data for the filesystem is requested from the disk, but is not yet available. What do device drivers generally do in this scenario?

---

7. Describe how I/O buffering can be formulated as a *bounded-buffer producer-consumer problem*.

---

8. An example operating system runs its interrupt handlers on the kernel stack of the currently running application. What restriction does this place on the interrupt handler code? Why is the restriction required?

### Multi-processors

9. What are the advantages and disadvantages of using a global scheduling queue over per-CPU queues? Under which circumstances would you use the one or the other? What features of a system would influence this decision?

---

10. When does spinning on a lock (busy waiting, as opposed to blocking on the lock, and being woken up when it's free) make sense in a multiprocessor environment?

---

11. Why is preemption an issue with spinlocks?

---

12. How does a read-before-test-and-set lock work and why does it improve scalability?

---

13. Describe how an MCS lock further improves scalability compared to a read-before-test-and-set lock? What is a disadvantage of an MCS lock?

---

## Scheduling

14. What do the terms *I/O bound* and *CPU bound* mean when used to describe a process (or thread)?

---

15. What is the difference between cooperative and pre-emptive multitasking?

---

16. Consider the multilevel feedback queue scheduling algorithm used in traditional Unix systems. It is designed to favour IO bound over CPU bound processes. How is this achieved? How does it make sure that low priority, CPU bound background jobs do not suffer starvation?

    Note: Unix uses low values to denote high priority and vice versa, 'high' and 'low' in the above text does not refer to the Unix priority value.

---

17. Why would a hypothetical OS always schedule a thread in the same address space over a thread in a different address space? Is this a good idea?

---

18. Why would a round robin scheduler NOT use a very short time slice to provide good responsive application behaviour?

---

19. Consider 3 periodic tasks with execution profiles shown below. Draw scheduling diagram for a rate monotonic and a earliest deadline first schedule.

| Process | Arrival Time | Execution Time | Deadline |
|---|---|---|---|
| A (1) | 0 | 10 | 20 |
| A (2) | 20 | 10 | 40 |
| | | . . . | |
| B (1) | 0 | 10 | 50 |
| B (2) | 50 | 10 | 100 |
| | | . . . | |
| C (1) | 0 | 15 | 50 |
| C (2) | 50 | 15 | 100 |

---

*Page last modified: 8:02am on Tuesday, 23rd of April, 2019*

Screen Version