109370210 工管四乙 黃鈺凱

OS HW #3

7.8

Because acquiring a semaphore may put the process to sleep while it is waiting for the semaphore to become available. Spinlocks are to only be held for short durations and a process that is sleeping may hold the spinlock for too long a period.

8.20

(a) Increase Available: This could safely be changed without problems

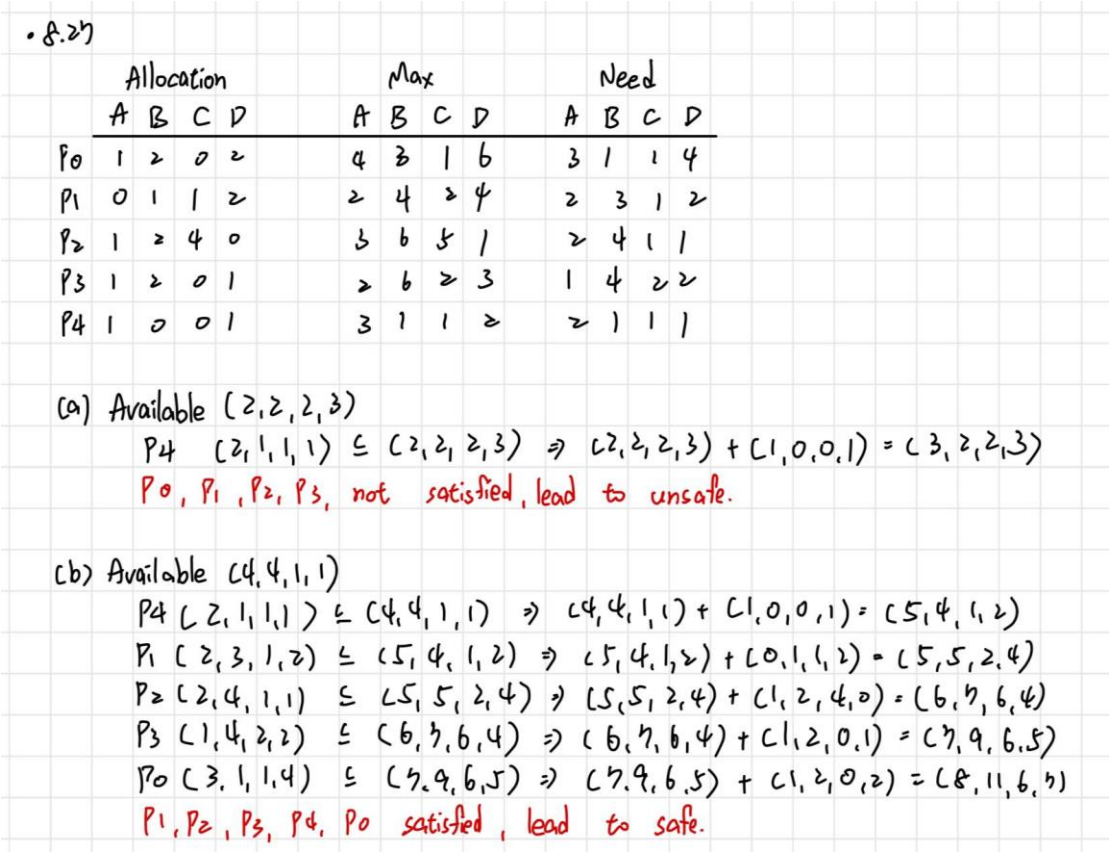
(b) Decrease Available: This could have an effect on the system and in roduce the possibility of deadlock as the safety of the system assumed there were a certain number of available resources.

(c) Increase Max for one process: This could have an effect on the system and Introduce the possibility of deadlock.

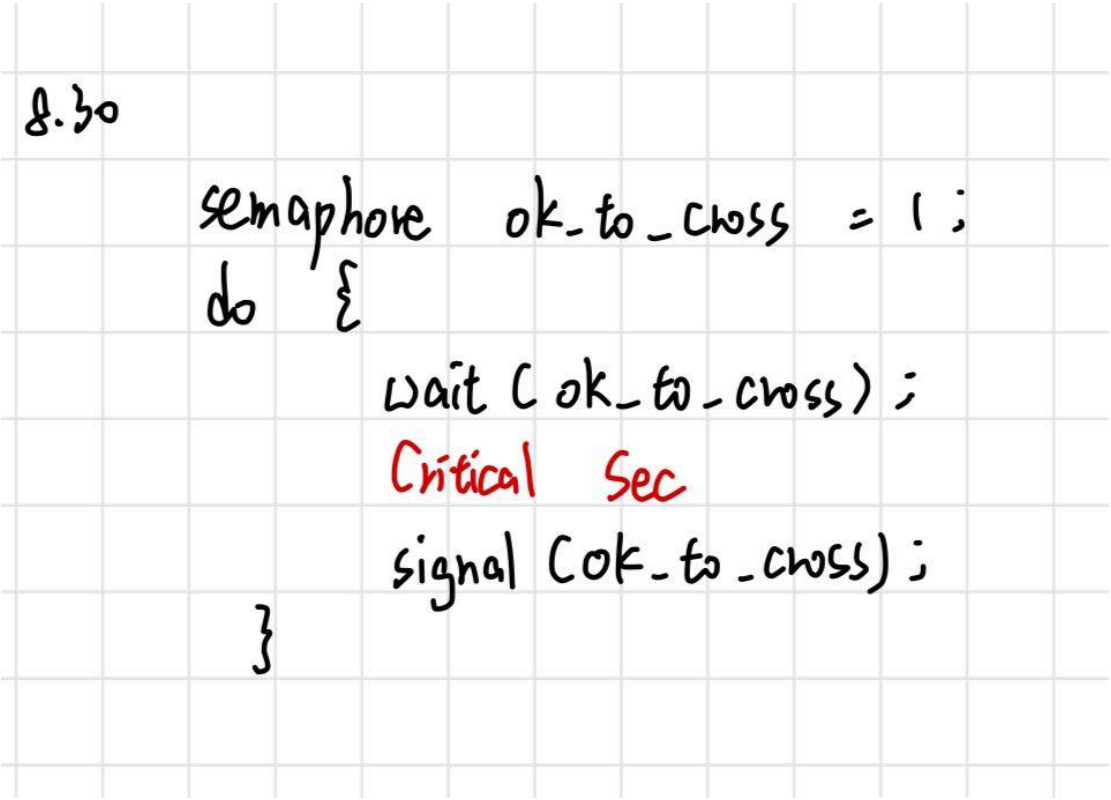
(d) Decrease the number of process: This could be allowed assuming that The system doesn't enter an unsafe state.

(f) Decrease the number of process: This could safely be changed without poblems.

8.27



8.30



9.15

(a) External Fragmentation:

Contiguous Allocation with fixed-sized partition:donot suffer from external fragmentation.

Contiguous Allocation with variable-sized partition:suffer from external fragmentation.

Pure segmention: suffer from externel fragmentation.

Paging: not suffer from external fragmentation.

(b) Internel Fragmentation:

Contiguous Allocation with fixed-sized partition: suffer from external fragmentation.

Contiguous Allocation with variable-sized partition:do not suffer from external Fragmentation.

Pure segmention:do not suffer from Internel fragmentation.

Paging: suffer from Internal fragmentation.

(C) Ability to share code across process.

Contiguous Allocation with fixed-sized partition:no support for code sharing across processes.

Contiguous Allocation with variable-sized partition:no support for code sharing across processes.

Pure segmention:support for code sharing across processes. However, must becareful to make sure thatprocesses do not mix code and data in the same segment.

Paging:support for code sharing across processes. However, must becareful to make sure thatprocesses do not mix code and data in the same page.

9.24

a. Conventional, single-level page table:

In a conventional page table, each page entry corresponds to a page in memory.
Number of entries = Number of pages = Physical memory size / Page size
Number of entries = (1 * 1024 * 1024 KB) / (8 * 1024 bytes)
Number of entries = 128, meaning there are 128 entries in the conventional, single-level page table.

b. Inverted page table:

In an inverted page table, each entry corresponds to a frame in memory.
Number of entries = Number of frames = Physical memory size / Page size
Number of entries = (1 * 1024 * 1024 KB) / (8 * 1024 bytes)
Number of entries = 128, meaning there are also 128 entries in the inverted page table.

So, both a conventional, single-level page table and an inverted page table in this scenario would have 128 entries each.