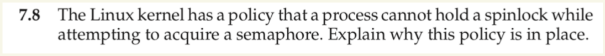
2025/04/28(一) 7.8 8.20、8.28(a)(d)、8.30 9.11 、9.13 9.14 111590004 張意昌



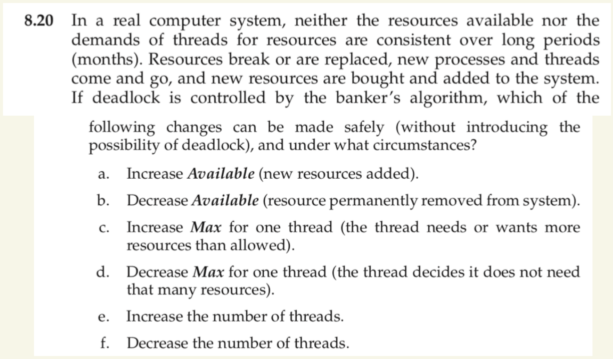
7.8 The Linux kernel has a policy that a process cannot hold a spinlock while attempting to acquire a semaphore. Explain why this policy is in place.

spin lock 是拿來 busy waiting 的。

Semaphore 是允許睡眠的同步工具。

* 你不能同時睡眠又 busy waiting

spin lock 不允許睡眠



8.20 In a real computer system, neither the resources available nor the demands of threads for resources are consistent over long periods (months). Resources break or are replaced, new processes and threads come and go, and new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

a. Increase ***Available*** (new resources added).

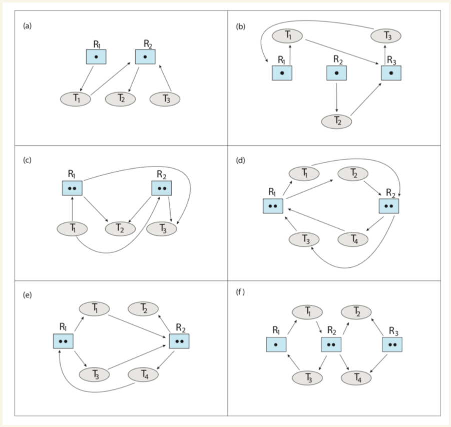
b. Decrease ***Available*** (resource permanently removed from system).

c. Increase ***Max*** for one thread (the thread needs or wants more resources than allowed).

d. Decrease ***Max*** for one thread (the thread decides it does not need that many resources).

e. Increase the number of threads.

f. Decrease the number of threads.



(a) 安全

因為增加可用資源會使 OS 工作更輕鬆。

不會進入不安全狀態。

(b) 不一定安全

要進行安全性檢查確認資源是否移除後是

Safely state，否則 Unsafe.

(c) 不一定安全

要進行安全性檢查確認添加後是否是安全的

Max > Avalible → 不安全

Max < Avalible → 不一定，要檢查

(d) 一定安全

移除最大資源要求使 OS 工作量減少，變輕鬆!

不進入 unsafe state。

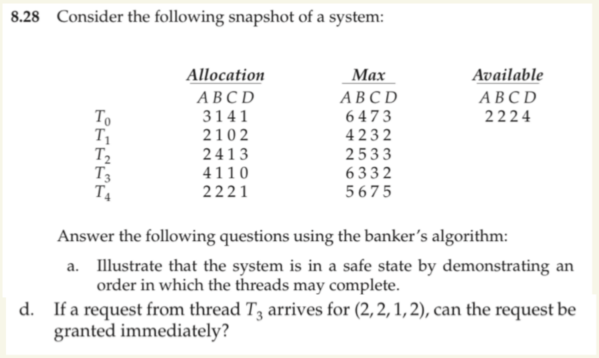
(e) 不一定安全。

增加 thread 提高需求，得進行 safety check

才能確保安全。

(f) 一定安全。

移除 thread 使資源需求量減少，肯定安全。



8.28 Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
T_0	3 1 4 1	6 4 7 3	2 2 2 4
T_1	2 1 0 2	4 2 3 2	
T_2	2 4 1 3	2 5 3 3	
T_3	4 1 1 0	6 3 3 2	
T_4	2 2 2 1	5 6 7 5	

Answer the following questions using the banker's algorithm:

a. Illustrate that the system is in a safe state by demonstrating an order in which the threads may complete.

d. If a request from thread T_3 arrives for (2, 2, 1, 2), can the request be granted immediately?

(a) Need : Max-Allocation seq $\langle T_2, T_0, T_1, T_3, T_4 \rangle$ ✓

× T_0 (3,3,3,2)

× T_1 (2,1,3,0)

× T_2 (0,1,2,0)

T_3 (2,2,2,2)

T_4 (3,4,5,4)

第一輪：T_2 → ok

Available (2,2,2,4) → (4,6,3,7)

Available += Allocation_{t_2}

第二輪：T_0 → ok Available (4,6,3,7) → (7,7,7,8)

第三輪 T_1 → ok Available (7,7,7,8) → (9,8,7,10)

第四輪 T_3 → ok Available (9,8,7,10) → (13,9,8,10)

第五輪 T_4 → ok Available (13,9,8,10) → (15,11,10,11)

(d)

給定 $T_3\ (2,2,1,2)$

$Req_{T_3} <= Need_{T_3}\ (2,2,1,2) \le (2,2,2,2)$ ✓

$Req_{T_3} <=$ Avaliable $(2,2,1,2) \le (2,2,2,4)$ ✓

試分配

Available $\Rightarrow (0,0,1,2)$

$Need_{T_3} \Rightarrow (0,0,1,0)$

$Allocation_{T_3} \Rightarrow (6,3,2,2)$

$\Downarrow$

safety check

第一輪 T_3 Available $(6,3,3,4)$

之後 T_0, T_1, T_2, T_4 依序檢查完

Seq : $\langle T_3, T_0, T_1, T_2, T_4 \rangle$

8.30 A single-lane bridge connects the two Vermont villages of North Tunbridge and South Tunbridge. Farmers in the two villages use this bridge to deliver their produce to the neighboring town. The bridge can become deadlocked if a northbound and a southbound farmer get on the bridge at the same time. (Vermont farmers are stubborn and are unable to back up.) Using semaphores and/or mutex locks, design an algorithm in pseudocode that prevents deadlock. Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge, or vice versa).

(a) mutex

```
mutex lock
procedure northfarmer():
    acquire (lock)
    walk ()
    release (lock)
procedure southfarmer():
    acquire(lock)
    walk()
    release(lock)
```

(b) Semaphore

```
Semaphore bridge=1
procedure northfarmer():
    wait(bridge)
    walk()
    signal(bridge)
procedure northfarmer():
    wait(bridge)
    walk()
    signal(bridge)
```

9.11 Explain the difference between internal and external fragmentation.

External：總空間足夠，只是可用空間在分配時不連續，無法滿足需求。

Internal：在請求空間時，實際要求與被分配到的空間的差值，分配＞實際要求。

9.13 Given six memory partitions of 100 MB, 170 MB, 40 MB, 205 MB, 300 MB, and 185 MB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 200 MB, 15 MB, 185 MB, 75 MB, 175 MB, and 80 MB (in order)? Indicate which—if any—requests cannot be satisfied. Comment on how efficiently each of the algorithms manages memory.

可用記憶體片段

A 100 B 170 C 40 D 205 E 300 F 185

分配 process

200 15 185 75 175 80

first-fit

200 → D (5)
15 → A (85)
185 → E (115)
75 → A (10)
175 → F (10)
80 → B (90)

best-fit

200 → D (5)
15 → C (25)
185 → F (0)
75 → A (25)
175 → E (125)
80 → B (90)

worst fit

200 → E (100)
15 → D (190)
185 → D (5)
75 → F (110)
175 → 沒有
80 → B (90)

best-fit 六個區段都用到
利用率好。

first-fit 用到部分區段但
有用乾淨。(都有分完)

worst-fit 用到部分但沒
分完 (效率最差)。

9.24 Consider a computer system with a 32-bit logical address and 8-KB page size. The system supports up to 1 GB of physical memory. How many entries are there in each of the following?

a. A conventional, single-level page table

b. An inverted page table

a. 8-KB 1KB 是 1024 bytes (2^{10})

$\Downarrow$

$8 \times 2^{10} \Rightarrow 2^{13}$ 13-bit

$2^{32}/2^{13} = 2^{19} \Rightarrow$ 有 2^{19} 個 entries

b. $1GB \Rightarrow 1024MB \Rightarrow 2^{10} \times 2^{10} KB \Rightarrow 2^{30}$ Bytes

$8\text{-}KB = 2^{13}$ bytes

$2^{30}/2^{13} = 2^{17}$ 個 entries #