(裝　訂　線)

國立成功大學　學年度第　學期第　次平時考試試卷

績成閱評		章簽師教		學生	院系	工學院 工科系	班	科目	名稱	作業系統
					學號	E94936116			班別說	工科系	班
					姓名	沈助宇					

1. False, not prevents deadlock. ✓
2. True ✓
3. False, 會 undo ✓
4. False, 因為會造成速率降低, 所以大部份 os 採取忽略死結, 由設計者解決
5. True, ✓

+10

Part II.

1
1. First come first served ✓
2. Shortest job first ✓
3. Round-Robin ✓
4. Memory management unit ✓
5. Translation look-aside buffer ✓

+5

2. (1) 若多個行程同時 load 一個資源, 即產生 load sharing, 此時要注意同步的問題 ✗ +2

(2) 當有行程在臨界區中執行, 則其他想進入的行程會在入口處持續執行迴圈等待進入, 即稱為 busy waiting ✓ +3

(3) 在一個 atomic transactions 的問題中, 若有問題產生, 則我們會去找 Log 的資料確認, 視情況做 redo or undo, 設立一個 checkpoint, 可讓我們不需回復 checkpoint 之前的正確狀態, 而直接從 checkpoint 後重新執行即可 ✓ +3

(4) 若有多個行程想共用同一個資源, 為了確保執行的順序正確, 我們給一個 timestamp 來記錄正確的時間, 如此才不會因順序更改造成錯誤

P_0	P_1
read(A)	
	read(A)
write(B)	
	write(B)

✓ +3

(5) 可分成計數器 semaphore 和二元 semaphore 兩種, 是一種方便的同步處理, 一次只能有一個行程進入 semaphore, 其他想進入 semaphore 的行程就必需等待, 確保資料同步 ✓ +3

3. 假如有三個行程 ABC, 其優先權順序分別是 A>B>C. 現在若 A 需使用的資源被 C 所占用, 則 A 必須要等待, 但 B 的優先權又高過 C, 則 C 必須等 B 執行完才能執行 如此 B 比 A 早執行, 造成 priority inversion. ✓ +5

解決方式: 使用優先權繼承, 此時優先權 A=C>B. 則 C 執行完後 A 即可執行

(續高轉背頁)

Ïhý uú

4. 例如交互執行 counter++ 和 counter-- 兩指令.

counter++ 可看成
```
register1 = counter
register1 = register1 + 1
counter = register1
```

counter-- 可看成
```
register2 = counter
register2 = register2 - 1
counter = register2
```

+5

若其交互順序如下, 初始 counter 為 5

register1 = counter	register1 = 5
register1 = register1 + 1	register1 = 6
register2 = counter	register2 = 5
register2 = register2 - 1	register2 = 4
counter = register1	counter = 6
counter = register2	counter = 4

則結果和答案不符. (4+5)
⇒ 這種互相競爭, 且其結果和排列順序有關, 即為 race condition.

5. 此程式碼會造成, 假若兩個行程同時要求進入臨界區
則 flag[i] = flag[j] = TRUE, 則兩個行程會進入 while 迴圈,
產生無窮等待, 都沒有辦法進入臨界區執行

6.
```
void swap (boolean *A, boolean *B) {
Boolean temp = A;
      A = B;
      B = temp; }
```

```
void swap (boolean *a, boolean *b)
Boolean *temp = a;
   a = b
   b = temp; }
```

```
do {
   key = TRUE;
   while (key == TRUE)
   swap (key, lock);
[critical section]
   lock = false;
[剩餘區] }
   while(1);
```

+5

// lock 預設為 false

```
do {
   key = TRUE;
   while (key == TRUE)
   swap (key, lock);
[critical section]
   lock = false;
[剩餘區]
while (1)
```

```
while (key == TRUE)
swap (key, lock);
```

7.
```
monitor montior_A {
procedure body (enter) {
[critical section] }
procedure body (exit) {
[剩餘區] }
  }
while(1);
```

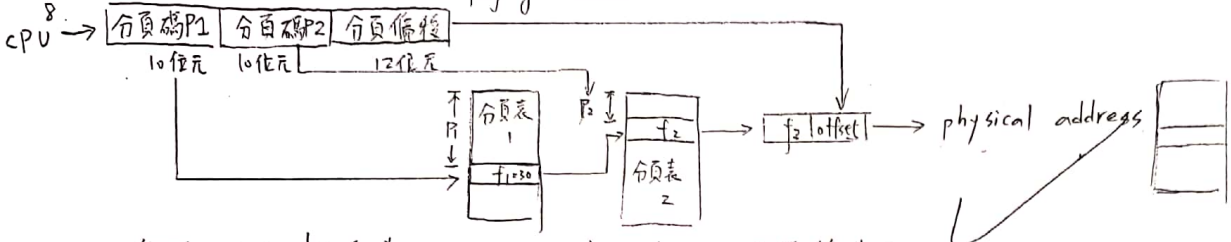
(若要產生區域變數
可用 condition x, y
再用 x.wait.
y.single 等方式實作) ✗ +3

(裝 訂 線)

國立成功大學 學年度第 學期第 次平時考試試卷

績成閱評		章教師教		學生	院系	工學院工科系97年級 班	科目	名稱	作業系統
					學號	E94936116		班開別設	工科系97年級 班
					姓名	洪功宇			

two level hierarchical paging scheme.



假設 page number 1 為 40. page numbe 2 為 50，頁偏移為 10
則到 page table 1 位址為 40 的地方，可得 30.
利用 30 這個值到 page table 2，則可以得到
最後的頁框 number = 400，則其頁框位址 = 400 + 100 = 500.
(physical address)

+10

9. (1) Need 為行程還需要的個數

	Need			
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2
	A	B	C	D

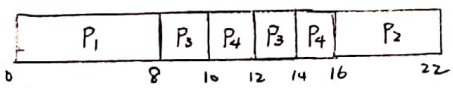
(2) Yes, 有一 safe 行到 $<P_0, P_3, P_4, P_1, P_2>$

(3)

	Allocation	Max	Need	Available
P_0	0 0 1 2	0 0 1 2	0 0 0 0	1 1 0 0
P_1	1 4 2 0	1 7 5 0	0 3 3 0	1 1 1 2
P_2	1 3 5 4	2 3 5 6	1 0 0 2	2 4 6 6
P_3	0 6 3 2	0 6 5 2	0 0 2 0	2 10 9 8
P_4	0 0 1 4	0 6 5 6	0 6 4 2	

有安全行到 $<P_0, P_2, P_3, P_4, P_1>$，不會產生死結.
所以 request 可以被執行.

10. 因為不可搶先



(1) P_1's waiting time = 0 ms (即 P_1 在等待行列花的時間)
(2) P_1's turnaround time = 8 ms (即 P_1 從開始到結束執行的時間)
(3) $\frac{4}{22} = 0.1818$ ⇒ 即每 ms (單位時間) 中，可執行 0.1818 個行程
(4) CPU 使用率，可視為 CPU 忙碌程度，0% 為最低，100% 為最高，實際約介於 40 ~ 70% 左右. X
(5) $\frac{0+16+(8+2)+(10+2)}{4} = 9.5$ ms X

+9

Ïhý uú