國立臺灣科技大學答案卷

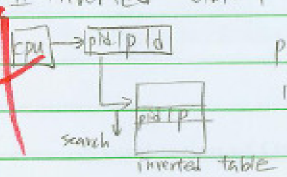
評分	教師簽名或蓋章
97	

科目 作業系統　任課教師 古鴻炎　系所班組別 四資工三

學號 B9415006　姓名 蘇品靜　考試日期 97 年 6 月 19 日

記分欄　從此處開始寫起，試卷用紙務須節用，非經主試認可不得續用其他紙張作答。

1a. 在 inverted table 中存放的 Data 有 pid 和 page number。　4



pid：這是屬於哪個 process 的

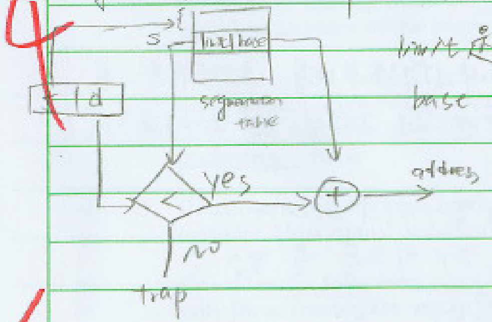
inverted 則是用 linear search 去比對 pid 和 p 的值，

∴ inverted 中會存有 pid 和 p

1b. Segmentation 的 table 中會存放 limit 和 base 二個值　4

limit 是指大小，他會和 d 做比較，d 不能大於 limit

base 是指起始位址，它會加上 d 成為真正的地址



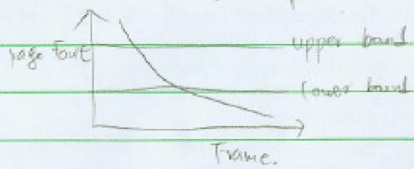
2.　6　1. Workset Model：在某個狀態時去紀錄所使用 page 數，藉此判斷是否可能發生 Thrashing

$D = \sum WSS_i$　當 $D > M$　D 是指每個 process 之前使用了哪些 page 的數量

超過 最大的page (M) [illegible] Thrashing,

若 D<M 。表示是安全的

2、page-fault Frequency：藉由判定upper bound和lower bound去判斷是否要加frame，避免Thrashing的發生



超過upper bound代表可能要加frame給它

低於lower bound代表frame數可能太多，可以拿掉一些

4

1、Demand paging的優點：① 通常在執行程式時，不太可能會同時用到所有的資源or程式

∴需要用到時才載入，可以節省memory accessing時間

② 而且也因為不用一下子載入那麼多東西，∴可以讓更多的程式進入執行的狀態，

throughput會上升。

③ response time也會下降

3、add C A B 假如A+B之後要放到C去，∴要抓A和B出來相加

① 第一個可能有page fault的情況 A不在memory上，∴發生page fault

② 第二個：B不在memory上，∴發生page fault

③ 第三個：C不在memory上∴也發生page fault

④ 若 add A, A,B A和B相加完要寫到A時發生page fault，此時A的內容已被更改了，∴不能回復重做。這種情形通常發生在來源和目地相overlap的時候。

97.3. 17,000 龍品

記分欄　轉頁從此開始寫起。

4. (a) FIFO

	1	3	2	4	1	6	3	1	2	5	1	4	1	5	2	3
1	1	1	4	4	4	3		3	3	1	1				1	3
3	3	3	3	1	1	1		2	2	2	4				4	4
		2	2	2	6	6		6	5	5	5				2	2

5

一共13次 page fault

(b) LRU

	1	3	2	4	1	6	3	1	2	5	1	4	1	5	2	3
1	1	1	4	4	4	3		3	5		5				5	5
	3	3	3	1	1	1		1	1		1				1	3
		2	2	2	6	6		2	2		4				2	2

5

一共12次 page fault

5a. 6

What is "race condition"? 就是指多個程式執行時因為共用一些變數，而又有中斷發生時，動作無法連續完成，可能造成執行結果有多種可能，無法確定真正結果是什麼，我們為 race condition。

What causes "race condition"? 二個程式有共用變數，且同時可能有多個程式 access，在任何寫入的指令中如果被打斷，就可能導致 race condition，出來的值無法確定。

ex: P1
do{
while(S>=0)

P2
do{
while(S>0

在 counter++ 和 counter--

counter++ counter--

}

5b. progress：代表這个 critical section 的後至少有一个 process 可以執行，不能多句因為你制定這个 critical section，就有一段時間沒人可以執行。程式的後一直有在前進有在接後執行才行

bounded waiting：是指不能讓有些 process 一直的進入 critical section，而有些程式一直都一直在等待，一直無法進入，必須讓每个 process 都有机會去執行到。

6a. TestAndSet(&lock) 用法大致如下：

```
do{
    value = TestAndSet(&lock)
        // do nothing
    // critical section
    lock = FALSE
}
```

把 lock 傳進去，只一直有人在用的話它會一直回傳1，而程式就一直不能執行到 critical section。

一直到沒有人在使用lock時，就會回傳0。

TestAndSet 本身有互斥的功能(mutual)，同時間只能讓一个人進入 critical section

6b.

```
do (semaphores *S) {
S->value --
if (S->value <= 0)
        remove one process from list
        wakeup()
}
```

→ 當沒有人被 lock 住，就把它 remove 出來

→ 並把它 from waiting → ready.

∴ 每次都會設定一个 process 來繼後執行

記分欄

轉頁從此開始寫起。

7

```
wait(mutex);
readcount++
if (readcount == 1) wait(wrt);
signal(mutex);
...
// reading is performed here;
...
wait(mutex);
readcount--
if (read count == 0) signal(wrt);
signal(mutex);
```

8a. 6

no preemption：就是資源不能被搶奪。即使能缺乏這項 resource，即別人有，也不能進行搶奪。

hold and wait：這個 process 本身已經持有一項 resource，但卻又在 wait 其它 resource.
(hold)

8b. 5

break circular wait 的方法如下：

把每一個 resource 做編號，並規定每一個 process 拿這些 resoura 時必須按順序拿

ex: process P_i 會用到 $F(P_i)=1$ 這3個 resource，它必須按順序拿，先拿1再拿2，才拿3 5，就不

F(P) = 2　　能用到2，作後是release 5 個以。

F(P) = 5　　這個方法有以下性質：F(Ri) < F(Ri+1)

又∵ F(R0) < F(R1) < ...... < F(Rn) < F(R0)

∴ F(R0) < F(R0) 是不可能發生的，∴ 可以 break circular wait。

8

9.

	Allocation A B C	Max A B C	Available A B C	Need A B C
P0	0 3 0	7 5 3	2 1 0	7 2 3
P1	3 0 2	3 2 2		0 2 0
P2	3 0 2	9 0 2		6 0 0
P3	2 1 1	2 2 2		0 1 1
P4	0 0 2	4 3 3		4 3 1

Banker's Algorithm of safety Algorithm

① work = 2 1 0 = available

Finish[i] = False

② Find i satisfy:

Need[i] ≤ work

Finish[i] = False.　if not find i go to step 4 ← check 有一個 list.

③ work = work + Allocate

Finish[i] TRUE

④ if Finish[i] = TRUE, system ⇒ safe

else ⇒ un safe

由 table 可知 Availble 是 2 1 0 並不能夠找到一個 list，∵ 2 1 0 不能執行 P3，P1 也不足夠，

找不到一個可執行的 sequence list，∴ System 是 unsafe 的

10、(a) 缺點：① 速度慢，用 link list 會導致速度下降　6

② 且 direct access 很不容易做到，幾乎是不太可能，∵ 要從頭到尾慢慢去找。

③ 不太可靠，if 中間某個 link 斷了，就會造成不知道 data 是誰的的 error。

(b) 缺點：① 會有 external fragmentation 內部碎片　6

② 也有 internal fragmentation 外部碎片

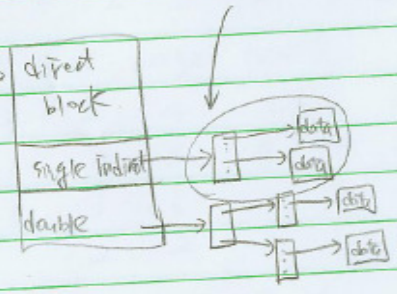
③ 如果無法一次 allocate 足夠大的空間，process 就得wait

可轉頁再寫。

記分欄 轉頁從此開始寫起。

11a. direct block：就是指此block含直接指到所要access的data去

single indirect：不含直接指到所要的data block去，含再透過一層的block去取access的記憶體



11b. FAT可以支援 direct access，它把所有的link list pointer都收集起來放在memory中，然後一下block一樣只需要index指到對應的block就可以，又因為在memory中執行，所以支援direct access。