作業說明

• 請在下列的70個問題中，挑選並完成其中20個題目。

• 其中每一章節需要完成至少1/5的題目(4, 2, 3, 2, 3題)。

• 其餘六題可以自行挑選要完成哪個章節的那些題目。

• 每一題請標明章節-題號(例如1-3, 5-4)並按照章節-題號順序作答。

• 請上傳完成的作業pdf檔案，可以使用手寫或打字。

• 可以參考投影片內容回答，但請不要直接複製投影片內容。

• 繳交截止時間2023/4/6 09:10。

Chapter 1

1. ■請描述一個作業系統應該具備的基本目標(goal)？

A:OS設計的基本目標應該要具備以下4種:

1.提供一個讓users易於操作電腦之溝通介面。

2. 提供一個讓users programs易於執行之環。

3.作為一個resource的管理者，協調分配resource，使resource可以有效利用。

4. 作為一個監督者，監控所有process執行，避免process之有意or無意的破壞，使system有重大危害。

1. ■在電腦系統架構中，一共包含四個基本元素：硬體(Hardware)、作業系統(Operating System)、應用程式(Application)、使用者(User)，請描述這四者間的基本關係。

A: 使用者(User)對應用程式(Application)下命令，應用程式(Application)利用system call發中斷請求給OS，並由OS分配硬體(Hardware)資源完成任務。

1. ■請解釋系統程式(System Program)和應用程式(Application Program)之間 的差別。

A: 系統程式(System Program)是指附帶(ships)於OS，但不屬於kernel的程式，通常被預先安裝在OS中，例如:Compiler、Assemble、Linking Loader、Debugger…。

而應用程式(Application Program)則是與OS分離的程式，例如:Office、Word、Processer…。

1. ■請解釋在電腦系統中，bus的意義。

A: Bus，即匯流排，是指資料傳輸的通道，它可以讓不同的硬體元件（例如CPU、記憶體、顯示卡、硬碟）之間透過共同的通道進行資料的傳輸。

1. ■請解釋甚麼是中斷向量表(Interrupt vector)

A: 中斷向量表(Interrupt vector)是指，在Interrupt I/O的運作流程中，當I/O-operating完成，Device controller會發出”I/O-completed”interupt通知CPU，CPU偵測到中斷，確認中斷發生種類，會跳到中斷向量表(Interrupt vector)中所對應的ISR或Interrupt handler之memory address去執行ISR；所以中斷向量表(Interrupt vector)中的欄位需要有中斷編號、ISR(或Interrupt handler)之memory address…

Chapter 2

1. ■請解釋何為CLI，以及它的優點及缺點。

A: CLI(Command-Line Interface)，是指使用者操作介面以命令列的方式呈現，優點:效能較好、較精確，缺點:介面較難被接受

1. ■請解釋何為GUI，以及它的優點及缺點。

A: GUI (Graphics User Interface)，是指使用者操作介面以圖形化的方式呈現，優點:介面較美觀、簡潔、易於使用，操作直觀，缺點:開發較費時、效能比較差

Chapter 3

1. ■對於一個執行中的程序，其記憶體部分可以分為text section / Stack / Data / Heap，請分別說明這些區域中存放了何種資料。

A: text section即code section，存放程式碼

Stack中存放該process的local variables、parameters、return address

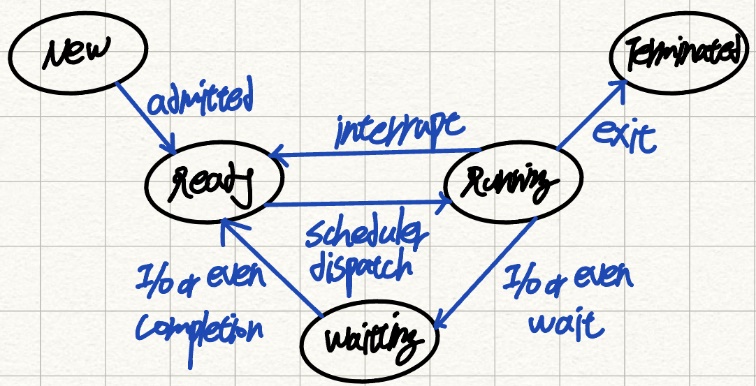
Data中存放該process的global variables(有初值與未設初值)  
Heap中存放向系統要求的額外配置之Memory space(例：pointer變數透過malloc new要求之空間)

1. ■請說明關於行程的五種狀態的名稱，以及五種狀態之間的關係。

A:

|  |  |
| --- | --- |
| state | explain |
| New(Create) | Process被建立，已分得PCB空間，尚未Loading進記憶體(or尚未取得記憶體資源) |
| Ready | Process已在記憶體中，且在Ready Queue內，具有資格爭奪CPU |
| Running | Process取得CPU執行中 |
| Wait(Block) | Process在Waiting Queue中wait for I/O completed or event occurs(不會與其他Process爭奪CPU) |
| Exit(Terminal、Zombie、Abort) | Process完成工作，正常結束或異常終止，此時可能其PCB尚未回收，因為要等其Parent Process collect 該subprocess的成果後，才會回收PCB空間，而其他resource已回收 |

|  |  |
| --- | --- |
| Transition | explain |
| Admit | 當記憶體空間足夠時，可由Long-Term Scheduler (in批次系統中)決定將此Job Loading進記憶體中 |
| Dispatch | 由Short-Term Scheduler(CPU Scheduler) 決定讓高優先權Process取得CPU |
| Time-out | 也稱Interrupt，執行中的Process會因為某些event發生，而被迫釋放CPU，回Ready Queue，例：Time-out、Interrupt、高優先權Process到達、插隊 |
| Wait | Wait for I/O completed或event occurs |
| I/O-completed | I/O-completed 或event occurs |
| Exit | Process完成工作或異常終止 |



1. ■請說明什麼是PCB，以及一個PCB中會包含哪些內容。

A: PCB(Process Control Block)OS為了管理所有Process，會在kernel memory中，替每一個Process各自準備一個Block(表格) ，記錄Process之所有相關資訊。PCB主要內容有：

1. Process ID：每個Process具有一個唯一的(unique)ID
2. Process state：Ready、Running、Waiting…
3. Programming Counter：內放下一個要執行的instruction’s address
4. CPU Register：紀錄使用到暫存器的值，例：Accumulator、password、Stack、Top
5. CPU排班資訊：紀錄Process priority、Arrived time、CPU time、quantum…
6. Memory Management information：隨OS記憶體管理方法不同，記錄不同資訊。例：Base/limit Register或Page Table或Segment Table
7. Accounting information：紀錄Process已使用多少CPU time、使用哪些資源、還剩多少CPU time可用…
8. I/O status information：紀錄Process已發出多少I/O-request、完成狀況如何、目前暫用哪些I/O資源…

Chapter 4

1. ■請解釋Process和Thread之間的關係。

A: Thread又稱”Lightening Process”，是OS分配CPU Time的對象單位，”Process”和”Thread”之間的關係好比”可行駛的汽車”與”引擎”的關係，每個Process至少存在一條Thread，好比每台可行駛的汽車至少存在一顆引擎。

Process VS Thread

|  |  |
| --- | --- |
| Heavy Weight Process | Lightening Process |
| Single-Threaded Model | Multithreaded Model |
| 是OS分配Resource的對象單位 | 是OS分配CPU Time的對象單位 |
| 不同Process不會有共享的Memory與其他Resource(除非採用shared memory) | 同一Process內之Thread彼此共享Process之Memory與其他Resource |
| 若Process內的single Thread被Blocked，整個Process亦被Blocked | 只要Process內尚有available Thread可執行，整個Process不會被Blocked |
| Process之Creation、Context Switching慢，管理成本高 | Thread之Creation、Context Switching快，管理成本低 |
| 對於Multiprocessors架構之效益發揮較差 | 對於Multiprocessors架構之效益發揮較佳 |
| Process無此議題 | 因為同一Process內之Thread彼此共享Process Data section，所以必須對共享的Data提供互斥存取機制，防止race condition |

1. ■請描述衡量一個工作的可平行度的Amdahl’s Law，並說明公式中每個部分的意義。

A:，其中S代表serial portion，N代表Processing croes數量，此公式是在計算在N個Processing croes下，平行化程度的比例對執行速度的影響，其中由公式得知，，代表真正影響平行化效率的因素不在Processing croes數量，而是在於平行化的程度多寡。

Chapter 5

1. ■請說明CPU排程器介入排程的四種時機點。

A: CPU Scheduler即為Short-Term Scheduler(短期排班器)，目的是從Ready Queue中，挑出一個高優先權Process並分派CPU給其執行。而CPU Scheduler有4個介入的時機點:

1. RunningWait (Wait)
2. RunningReady (Time out)
3. WaitReady(I/O completed、Wake up)
4. Terminat (Exit)

其中的一跟四在排程時，是不能中斷的，二跟三是可以中斷，但需要考慮到是否有shared data、是否在kernel mode，還有他們是不是真的能中斷。

1. ■請說明Preemptive和Nonpreemptive的主要差別。

A:

|  |  |  |
| --- | --- | --- |
|  | Nonpreemptive | Preemptive |
| 定義 | 除非執行中的Process自願放掉CPU，其他Process才有機會取得CPU，否則就只能wait，不可逕自搶奪CPU | 執行中的Process有可能被迫放棄CPU回到Ready Queue，將CPU切給別的Process使用 |
| 優點 | 1. Context switching次數較少 2. Process完工時間點較可以預期(predictable) 3. 比較不會產生Race condition | 1. 排班效能較佳，平均waiting、turnaround time較小 2. 適合用在Time-sharing System與Real-Time System |
| 缺點 | 1. 排班效能較差(可能有Convoy Effect) 2. 不適合用在Time-sharing System與Real-Time System | 1. Context switching次數較多 2. Process完工時間較不可預期 3. 需注意Race condition發生 |

1. ■請解釋Dispatcher的功用及作業流程。

A: Dispatcher(排程器)，是在CPU排程功能中的一個重要元件，Dispatcher是一塊Module，由他把CPU的控制權授予經由Short-Term Scheduler(短期排班器)所選出的Process。Dispatcher的功能包括：

1. Switching context
2. Switching to user mode
3. Jumping to the proper location in the user program to restart that program

上述3個工作的時間加總，即為Dispatch latency(分派延遲)。

1. ■在CPU排程中，其中一種衡量的指標是Response Time，請說明Response Time的定義。

A: Response Time(回應時間)即為user(user program)input命令/Data 給系統到Systeme產生第1個回應的時間差。

1. ■在CPU排程中，其中一種衡量的指標是Waiting Time，請說明Waiting Time的定義。

A: Waiting Time(等待時間) 即為Process花在Ready Queue中等待獲得CPU之等待時間加總。

1. ■請說明排程演算法中FCFS(First-Come-First-Service) Scheduling的運作 方式。

A: FCFS的做法是：到達時間最小的Process會最優先取的CPU。FCFS的分析如下:

1. 排班效能最差，即average waiting time、average turnaround time最長，但製作簡單(使用Ready Queue即為FCFS)
2. 可能有”Convoy Effect”(護衛效應)

(護衛效應:許多Process均在等待一個需很長CPU Time之Process完成工作，才能取得CPU，造成average waiting time痕長之不良現象)

1. 公平
2. No Starvation(沒有飢餓現象)
3. Non-Preemptive法則(不可插隊、不可搶先)

例:P1、P2、P3依序且近乎同時到達

|  |  |  |
| --- | --- | --- |
| Process | CPU(burst)Time |  |
| P1 | 24 |
| P2 | 3 |
| P3 | 3 |

1. ■請說明排程演算法中SJF(Shortest-Job-First) Scheduling的運作方式。

A: SJF的做法是：具有最小時間的CPU Time之Process優先取得CPU。SJF的分析如下:

1. 排班效益最佳(optimal)，即average waiting time、average turnaround time最小(Why?因為Short Job所減少的waiting time必定大於等於Long Job所增加的waiting time，使average waiting time為最小)
2. 不公平，偏好Short Job
3. 可能會Starvation(for Long Jobs)
4. 可分成
   * + 1. Nonpreemptive:以SJF做代表
       2. Preemptive:另外稱作SRTF
5. 較不適合用在Short-Term Scheduler

例1:P1、P2、P3、P4依序且近乎同時到達

|  |  |  |
| --- | --- | --- |
| Process | CPU(burst)Time |  |
| P1 | 8 |
| P2 | 6 |
| P3 | 7 |
| P4 | 3 |

例2:P1、P2、P3、P4不同時到達

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrive Time | CPU(burst)Time |  |
| P1 | 0 | 7 |
| P2 | 2 | 4 |
| P3 | 4 | 1 |
| P4 | 5 | 4 |

1. ■在估計一個行程所需的執行時間時，可以使用指數平均(Exponential Averaging)的方式進行，請描述指數平均的計算方式。

A:(下次預估值=本次實際值+(1-加權值)\*本次預估值)，其中

1. ■請說明排程演算法中SRF(Shortest-Remaining-First) Scheduling的運作方 式。

A:SRTF(Shortest-Remaining Time First)即為”Preemptive SJF”法則，由剩餘的CPU burst Time 最小者取得CPU，也就是若新到達之Process，其CPU burst Time小於目前執行中Process的剩餘CPU Time，則新到達之Process可以插隊(Preemptive)執行。SRTF的分析如下:

1. 與SJF相比，SRTF之平均waiting、turnaround time會較小，但Context Switching的負擔較大
2. 不公平、偏好Short Running-Time Job
3. 會有Starvation
4. Preemptive法則

例:

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | CPU(burst)Time |  |
| P1 | 0 | 8 |
| P2 | 1 | 4 |
| P3 | 2 | 9 |
| P4 | 3 | 5 |

1. ■請說明排程演算法中RR(Round-Robin) Scheduling的運作方式。

A:OS會設定一個可參數化的CPU Time Quantum(or Slice)，當Process取的CPU執行後，若未能在此Quantum內完成工作，則Timer會發出Time-out Interrupt通知OS，OS會牆破此Process放掉CPU且回Ready Queue中，等待下一輪再取得CPU執行，每一輪之中，Process是以FCFS排班方式取得CPU。RR的分析如下:

1. Time-sharing System採用、是一個可參數化的法則
2. 公平(FCFS)
3. NO Starvation
4. Preemptive法則(Running to Ready)
5. RR Scheduling效益取決於Time Quantum大小之制定
6. ■請說明排程演算法中RR(Round-Robin)中的參數q如何影響排程。

A: 參數q即為Quantum值的大小，若q=無窮大，則RR會退化成FCFS法則，排班效能較差。若q=極小值，則Context switching太頻繁，CPU utilization趨近於0。依經驗法則，若Quantum值能讓80%的Jobs在Quantum內完工，效能較佳。

1. ■請說明排程演算法中PR(Priority) Scheduling的運作方式。

A: Priority Scheduling作法是讓具有Highest Priority之Process優先取得CPU，若多個Process權值相同，以FCFS為準。Priority Scheduling的分析如下:

1. 是一個具參數化的法則，即給予不同的Priority高低定義，可展現出不同的排班行為。例：

|  |  |  |
| --- | --- | --- |
| Priority定義 | 行為 | 包含於關係 |
| Arrived time越小，優先權越大 | FCFS | FCFSPriority |
| CPUtime越小，優先權越大 | SJF | SJFPriority |
| 剩餘時間越小，優先權越大 | SRTF | SRTFPriority |

1. 不公平
2. 會有Starvation(可用Aging解決)
3. 分為NonPreemptive與Preemptive 2種

例1:依序且近乎同時到達(越小的Priority Number有越高Priority)

|  |  |  |  |
| --- | --- | --- | --- |
| Process | CPU(burst)Time | Priority Number |  |
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

例2:

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Priority Number | CPU(burst)Time |
| P1 | 0 | 5 | 5 |
| P2 | 2 | 2 | 3 |
| P3 | 5 | 4 | 8 |
| P4 | 10 | 3 | 4 |
| P5 | 13 | 1 | 6 |
|  | | | |

1. ■請說明Multiple-Level-Feedback Queue的運作方式。

A:與Multilevel Queue相似，差別只在允許Process在不同Queue之間移動，所以可以採取類似”Aging技術”(優先權升高)：每隔一段時間，就將Process全部往上移一層Queue，所以在有限的時間之後，Lowest Priority Process會上升到hightest Priority Process

1. ■在負載平衡(load balancing)中可以進行的操作分為Pull-migration和Push-migration，請解釋兩者的差異。

A:實施Load Balance有兩個調節機制：

1. Push-migration(Top-down):

Kernel會定期確認各CPU的Work load，將loading重的CPU之Ready Queue內的一需Process到idle(或負擔較輕的)CPU之Ready Queue。

1. Pull-migration:

當CPU idle(loading低)，會發出請求給Kernel，請Kernel拉一些Process給該CPU執行。

由上述可知Pull-migration和Push-migration的差異在於CPU是主動還是被動的去要求Process。

1. ■在負載平衡(load balancing)中，親和性(affinity)會影響到負載平衡的操作，請說明為什麼。

A:Process Affinity是指在Multiprocessors System中，當Process已決定某CPU上執行，則在他執行的過程中，盡量不要將他轉移到其他CPU上執行(除非有必要，像是Processor BAD、load balancing…)，避免CPU內之cache等內容要複製(repopulation重新布置)且刪去(elimination)之影響工作效能。可分為1.Hard Affinity:規定Process不可轉移 ;2.Soft Affinity:盡可能不要轉移，若有需要仍可轉移。