1.

飢餓(Starvation)：一個process可能因長期無法取得資源而不能執行(並非完全無法取得，但一直沒有機會取得)

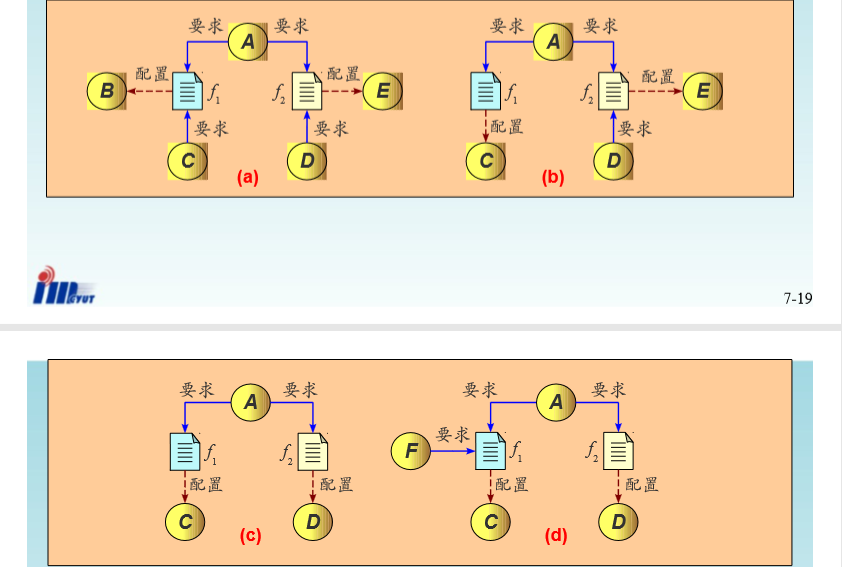
例如下圖：

(a)初始狀態：行程A同時要求資源f1及f2，行程B擁有f1，行程C 要求f1，行程D要求f2，行程E擁有f2

(b)B 結束並釋放f1，A無法使用f1因為f2無法取得 → f1配置給C

(c)E結束並釋放f2，A無法使用f2因為f1無法取得 → f2配置給D

(d)一個新的行程F可能開始執行而且要求f1 ... → A可能持續飢餓



1. SRJF (剩餘最短優先) preemptive 可搶先的SHF
2. FCFS (First-Come First-Served，先到先服務) non preemptive 先進先出的佇列，誰先進入等待佇列，誰就能先佔用CPU，直到行程結束或該行程執行等待I/O事件時才會釋放CPU使用權。
3. **SJF (Shortest Job First，最短工作優先) non preemptive 選擇執行時間最短的process先執行，但此種方法必須事先知道或是估算process所需的執行時間，如果行程長度相同，採用FCFS。**
4. Priority (優先權優先) preemptive 有高優先權的行程優先執行
5. Round-Robin non preemptive 時間量切成一段一段，每個process一次給一段

time slice 很長 -> FCFS

很短->processor sharing, more context switch

//Priority (優先權優先) 屬於可搶先跟不可搶先。

//可搶先的定義為非自願性被踢出 CPU。

<https://publish.get.com.tw/BookPre_pdf/51MG060707-2.pdf>

http://cc08310112tw.pixnet.net/blog/post/51441151-%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1%E7%AD%86%E8%A8%98-5.3-scheduling-algorithms

2. avoid starvation

a.老化機制(aging)：

給每個process一個年齡，代表process在就緒佇例中等待的時間；當年齡增加到一定程度時，就將process優先權提高。

b. Time quantum

OS規定一個CPU time quantum，若process在取得CPU後，未能於quantum內完成工作，則必須被迫放棄CPU，等待下一次輪迴。

3.

thread 是 OS 能夠進行運算排程的最小單位，它被包含在 process 之中，是 process 中的實際運作單位。

thread 是 OS 分配 CPU 時間的對象。

process 是 OS 分配資源的對象。

threads 彼此共享 :

* code Section
* data Section
* OS Resources

<http://mropengate.blogspot.tw/2015/01/operating-system-ch4-multithread.html>

4.

|  |  |
| --- | --- |
| Process | Thread |
| Heavy Weight Process | Light Weight Process |
| OS 分配資源的對象 | OS 分配 CPU 時間的對象 |
|
|
| Run in separate memory spaces | Run in a shared memory space |
| Single thread | Multithreads in the process |
|
| process與process無共享 | 必須提供對Shared Data的互斥存取控制，以防止不正常Thread存取所造成危害 |
|
|
|
|
|
|
|

http://www.csie.ntnu.edu.tw/~swanky/os/chap4.htm

5.

* short-term (CPU scheduler) —

從ready queue中挑選一個process來獲得CPU的使用權。

* medium-term —

用於分時系統的排程器。當記憶體不夠的時候，需要暫時將部分的執行程序搬出，將新來的程序載入，置換機制被用來移除部份在記憶體執行中的程式及恢復那些被暫時移除的程式。

* long-term (job scheduler) —

將 job 變成 process 的工作排程，將工作從硬碟載入到主記憶體中。determines which jobs are brought into memory for processing

6.

7.

Interrupt:

發生一個事件，通常是由硬體或軟體產生中斷來通知，當電腦的周邊設備有資料產生時，周邊設備會通知 CPU，而 CPU 會先擱下目前的工作並把當時的狀態存起來，然後處理這些資料，待處理完成後，再繼續被打斷的工作。

interrupt是由軟體所產生的插斷,，trap是由硬體所產生的系統流程的改變。

* Trap:

軟體產生

an exception in a user process.

1. User program need service to provide by OS
2. Catch up arithmetic error

divide-by-zero

* Interrupt:

硬體產生

something generated by the hardware.

1. IO-completed
2. Device-ready
3. I/O-error

* 兩者相同之處

從 User Mode 切換到 Kernel Mode。

兩者都算是 interrupt 中斷指令。

8. advantage of process cooperation

1. Information sharing
2. Computation speedup
3. Modularity 模塊化
4. Convenience

9. IPC (interprocess commutation)

1. Shared memory
2. Message passing

10.

* Context switch(環境切換):

當CPU從一個process切到另一個process執行之前，OS必須保存原有process的執行狀態，同時載入new process的執行狀態而 Context switch是一系統負擔，其時間長短，幾乎完全取決於Hardware因素

Ps.

降低負擔方法: 增加register、利用thread取代process、利用register

* System call:

User program向OS的kernal請求需要許可權運行的服務。

當user program需要OS提供服務時，藉由呼叫system call通知OS，OS可依據system call ID查表，啟動service routine執行，得到結果，再傳回給user program，完成服務請求。

* Virtual machine:

透過軟體模擬的技術，創造出一份與底層Hardware一樣的功能介面

* 利用CPU scheduling技術創造出有多顆CPU的假象
* 利用virtual memory技術創造出更多的記憶體空間
* 透過spooling技巧，提供多套I/O Device之功能

Ps.

優點：

* 作為OS測試、開發一個很好的平台工具
* 同一部Hardware上，可同時執行多套不同的OS，每個OS可Run在不同的virtual machine上
* Virtual machine具有保密性與安全性

缺點：

* 要創造出一部與底層Hardware一模一樣功能的Virtual Machine極為困難
* Virtual Machine之間的communication較為困難
* Batch:

將同類的工作集合起來一起執行，以減少重複且瑣碎的動作，並減少工作切換的時間，以提升系統效率。

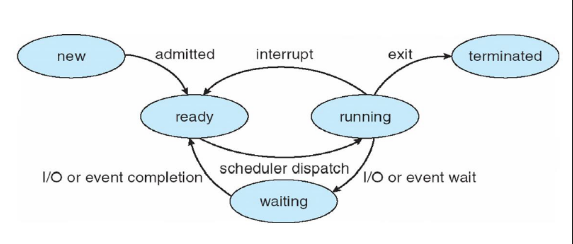
* Time sharing:

利用CPU scheduling 和multiprogramming技術將一份程式分配一個time quantum，即使沒有執行完，時間一到還是要交出控制權給其他使用者。

需要條件：要有interrupt的機制和timer來控制每個process的時間。

11.

12.

a.

<http://morris821028.github.io/2014/04/16/lesson/operating-system-notes/>

<http://wayne.cif.takming.edu.tw/os/os_5.pdf>

<https://sls.weco.net/node/21325>

http://cc08310112tw.pixnet.net/blog/post/51441151-%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1%E7%AD%86%E8%A8%98-5.3-scheduling-algorithms