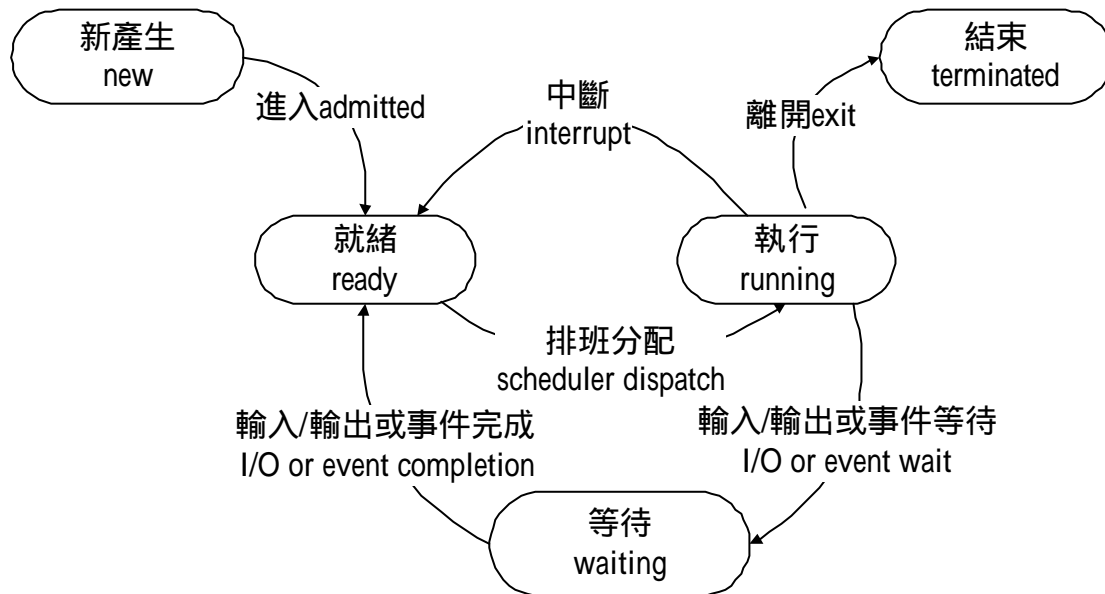




行程

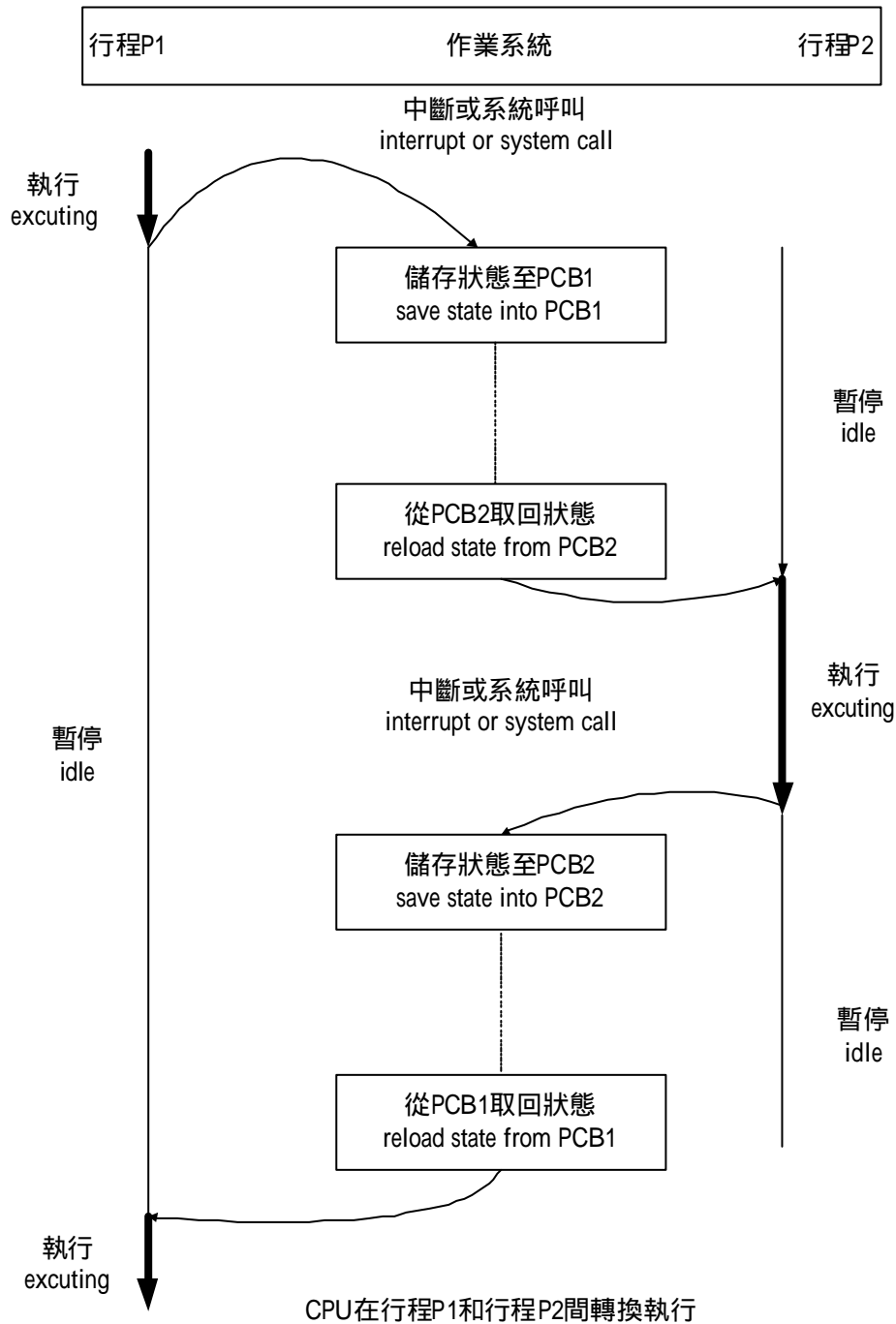
一個行程就是一個程式正在執行。當我們在終端機下達指令時，FreeBSD 就會建立一個行程，而當我們的程式執行完時，這個行程就被終止了。在一個分時系統的 FreeBSD 作業系統，允許多個使用者使用電腦系統，而許多行程也同步的被執行。像我們的個人電腦，一般只有一顆 CPU 中央處理器，但卻同時的處理多個行程，而這就分是分時系統。

當我們執行程式時，電腦作業系統就會使用 `new` 幫我們產生新的行程，而當我們的行程就緒(ready)時，而核心又排班分配 CPU 中央處理器給他時，他就會開始執行 running，直到執行結束 terminated(或 Zombie)。當然在過程中也有可能發生像系統呼叫的 System call 而發生中斷，或者改成其它行程執行(被 preempt)，這時我們的行程就會回到就緒 ready 的狀態。當然在過程中也可能發生像輸入/輸出 I/O 或事件等待(sleep)，此時，CPU 是在閒置 waiting 的狀態，而當我們的輸入/輸出或事件完成時(被 wake up)，才會到就緒 ready，等待下一次排班分配 CPU 中央處理器，直到結束(zombie 或 terminated)。



行程狀態圖

這是分時系統當行程 P1 和行程 P2 作執行上的內容轉換 Content switch。當行程 P1 在執行時，因為作業系統的中斷或系統呼叫，這時行程 P1 就要先將它的 Process Control Block(行程控制表 PCB1)給儲存起來。而這時後 CPU 中央處理器會載入所呼叫的行程 P2，這時會載入行程 P2 的 Process Control Block(行程控制表 PCB2)，這時 CPU 中央處理器就會執行行程 P2，而當執行行程 P2 一定的時間，又會將行程 P2 的 Process Control Block(行程控制表 PCB2)給儲存起來。而此時 CPU 中央處理器就會載入行程 P1，而執行行程 P1。



這就是行程狀態控制表。PCB(Process Control Block)含有行程的許多內部資訊。

行程狀態：可以是 new、ready、running、waiting 或 halted。

程式計數器：指明該行程下次要執行的指令位置。

CPU 暫存器：其數量和類別依不同形式 CPU 而有不同。有累加器(accumulators)、索引暫存器(index register)、堆疊指標(stack pointer)、位址暫存器(address register)....。

記憶體管理資訊：這包括存取記憶體的基底暫存器(base register)和限制暫存器(limit register)、分頁表(page table)或記憶體系統的分段表(segment table)的資訊。

CPU 排班法則和相關資訊：包括行程的優先順序、排班法則。

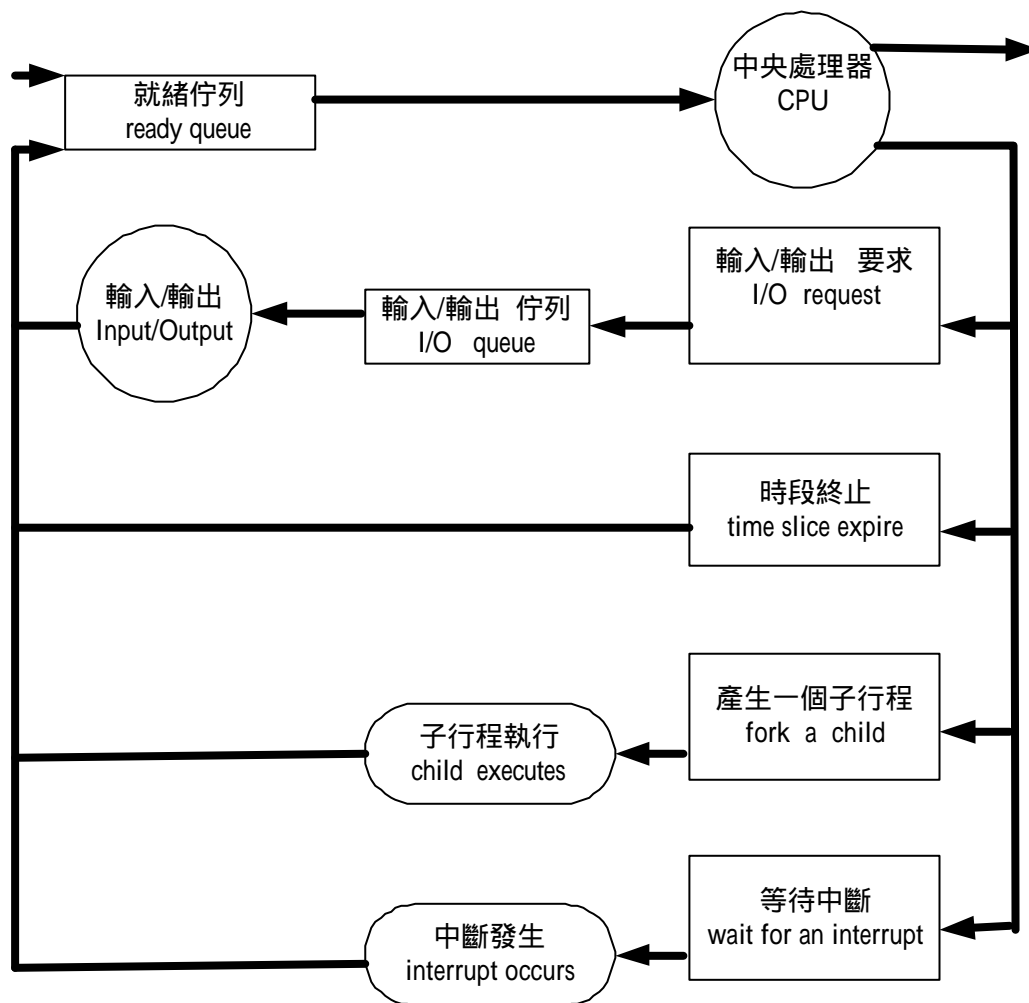
CPU 會計資訊：CPU 的使用時間、帳號、工作和行程的編號。

輸入/輸出狀態資訊：這包括了行程的輸入/輸出裝置、開啟檔案的串列。

指標pointer	行程狀態process state
行程號碼process number	
程式計數器program counter	
暫存器registers	
記憶體限制memory limits	
已開啟檔案表list of open files	
.	
.	
.	
.	
.	

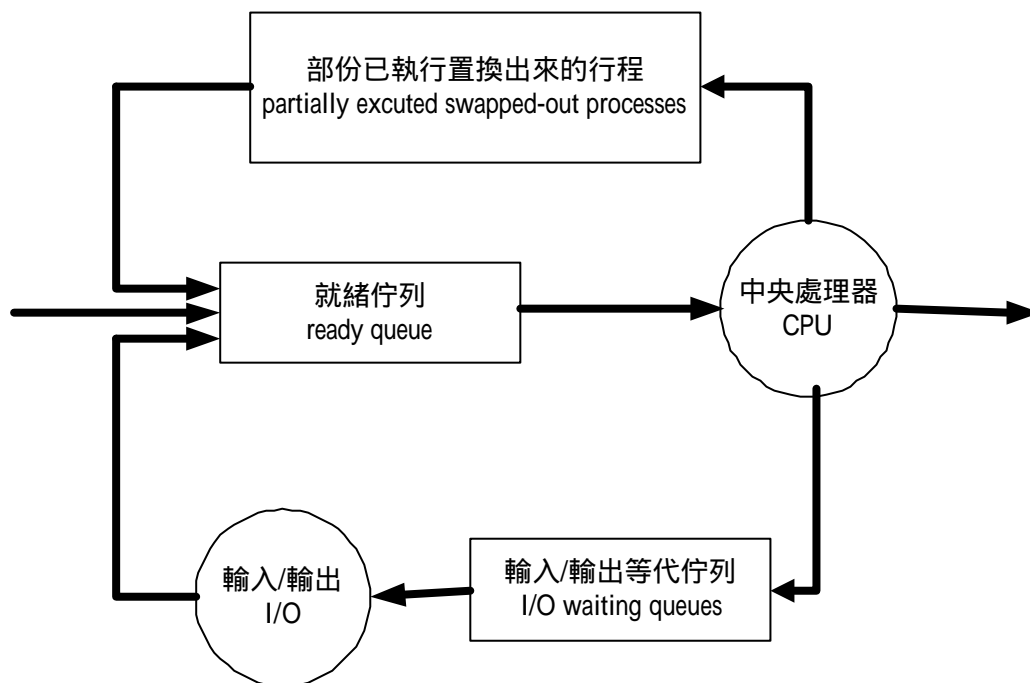
行程控制表PCB

分時系統的目的地是將 CPU 中央處理器在不同的行程之間不斷的轉換，以便讓使用者可以在自己的行程執行時與它交談。我們個人電腦的單一 CPU 處理器，不可能多個行程同時執行，而它們必需在一旁的佇列中等待，一直到 CPU 有空時，才可能排到它們執行。當我們新增 new 一個新的行程時，它最初是置於就緒佇列中。每一個長方形就是一個佇列，圓圈代表執行行程的資源，箭頭代表執行的方向。當我們所執行的程式或行程在進行時，它會先配置 CPU，然後有可能發出輸入/輸出的要求，然後置於一個 I/O 佇列中。程式也可能 fork 產生新的行程。行程也可強行的離開 CPU 然後就回到就緒佇列中。



行程排班佇列圖

在分時系統中一個行程在它整個生命期，將在各種不同的排班佇列中，這當然要看我們的需要設計，以及作業系統的設計。因此作業系統需按排班方式從佇列中選出行程，讓所選出的行程讓中央處理器 CPU 來執行。而我們所執行的程式也可能只執行一下子，然後就等待輸入/輸出 I/O 的要求，但 I/O 的時間，比 CPU 所需執行的時間大上很多。在整批作業系統中，大部份的行程都會以 spooling 儲存池的方式存到硬碟或大型儲存裝置，等待一個一個的執行。分時系統的中程排班程式是將行程從記憶體中移開，因此而降低多元程式規化的程度，然後再將行程放回記憶體中，並且放在它移開前的位置繼續執行，這種方法稱為 swapping，如下圖所示。



加入中程排班的佇列圖

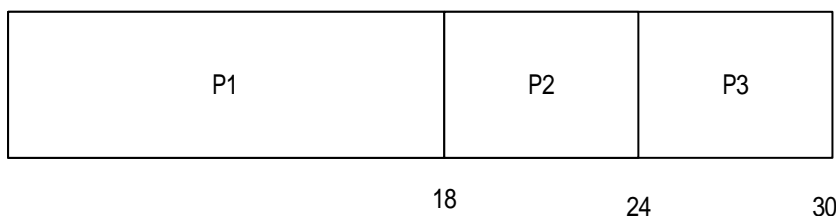
CPU 排班是多元程式規畫作業系統的基礎，藉由不同的行程在 CPU 執行中作轉換，讓我們電腦的輸出產量提高。我們在分時系統的作業系統 FreeBSD 中，CPU 一般都是可搶先排班(Preemptive Scheduling)，也就是當某個程式 P1 在執行時(某個行程在使用 CPU 資源)，此時，另外一個程式 P2 可以強奪 P1 程式 CPU 的資源，而此時則是 P2 程式在執行。我們有多種 CPU 排班法可以來決定是否使用這些排班，而它們評估的標準有 CPU 的使用率、CPU 的產量、每一個行程回復時間、行程提出要求到第一個反應出現所需的時間。

CPU 排班是決定將 CPU 分配給就緒佇列中的那一個行程，我們將介紹先來先做的排班方法、最短的工作先作的排班方法、優先順序的排班方法和依序循環的排班方法。

這是先來先作的排班法則，也就是 FCFS 的演算法，行程 P1 第一個到，因此先將 CPU 分配給它，等到 P1 行程執行完再執行行程 P2。P3 則是最後執行。當 P1 行程執行完 18 個單位時間，P2 才開始執行，此時 P2 已經等待了 18 個單位時間。同理，P3 等待了 24 個單位時間。因此平均等待時間為 $(18+24)/3=14$ 秒。這個圖又稱為甘特圖。

先來先作排班法 First Come, First Served Scheduling

行程	分割時間
P1	18
P2	6
P3	6



行程 P1 的等待時間是 0，行程 P2 的等待時間是 18，行程 P3 的等待時間是 24。因此平均等待時間是 $(0+18+24) / 3 = 14$ 。

這是我們最短的工作(行程 Process)先作的排班法 因為行程 P2 和行程 P3 所需分割時間同為 6，因此可以作為先分配 CPU。而行程 P1 所需分割的時間最長，所以最後才分配 CPU 給行程 P1，此時 P1 已經等待了 12 個單位的時間。因此平均行程所等待的時間 P1 為 12、P2 為 0、P3 為 6，所以平均等待的時間為 $(12+0+6)/3=6$ 。

最短的工作先作排班法 Shortest Job First Scheduling

行程	分割時間
P1	18
P2	6
P3	6

P2	P3	P1
----	----	----

6

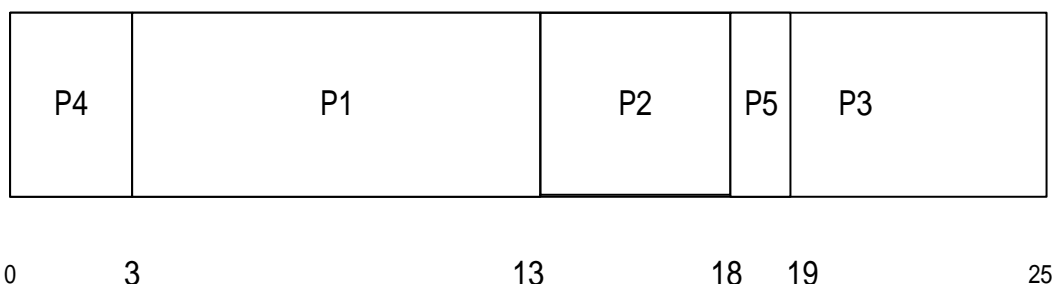
12

這是我們最短的工作(行程 Process)先作的排班法。因為行程 P2 和行程 P3 所需分割時間同為 6，因此可以作為先分配 CPU。而行程 P1 所需分割的時間最長，所以最後才分配 CPU 給行程 P1，此時 P1 已經等待了 12 個單位的時間。因此平均行程所等待的時間 P1 為 12、P2 為 0、P3 為 6，所以平均等待的時間為 $(12+0+6)/3=6$ 。

在優先權的排班法則中行程 P1 的優先權第 2，而它的等待時間是 3。行程 P2 的優先權是第 3，因此在甘特圖中它的等待時間是 13。行程 P3 的優先權是第 5，因此在甘特圖中，它的等待時間是 19。行程 P4 的優先權是第 1，因此等待時間是 0。行程 P5 的優先權是第 4，等待時間是 18。 因此平均等待時間是 $(3+13+19+0+18) / 5 = 10.6$ 。

優先權的排班法 Priority Scheduling

行程	優先順序	分割時間
P1	2	10
P2	3	5
P3	5	6
P4	1	3
P5	4	1

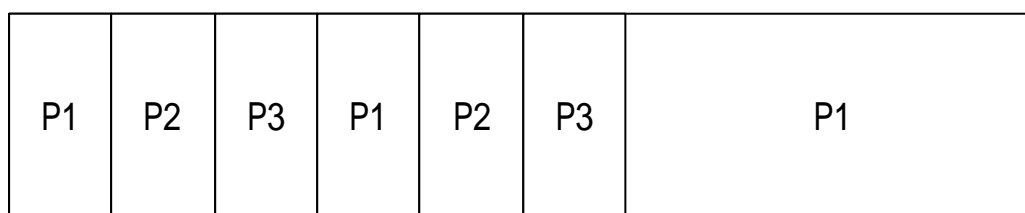


行程 P1 的等待時間是 3，行程 P2 的等待時間是 13，行程 P3 的等待時間是 19。行程 P4 的等待時間是 0，行程 P5 的等待時間是 18 因此平均等待時間是 $(3+13+19+0+18) / 5 = 10.6$ 。

這是我們 Round-Robin 依序循環的工作排班法，我們假設每次分配 CPU 給行程 3 個單位的時間。P2 在作完行程前總共等待 12 秒，P3 在作完行程前總共等待 15 秒，而 P1 在行程作完前總共等待 18 秒。因此平均等待時間是 $(12+15+18)/3=15$ 秒。

依序循環排班法 Round-Robin Scheduling

行程	分割時間
P1	18
P2	6
P3	6

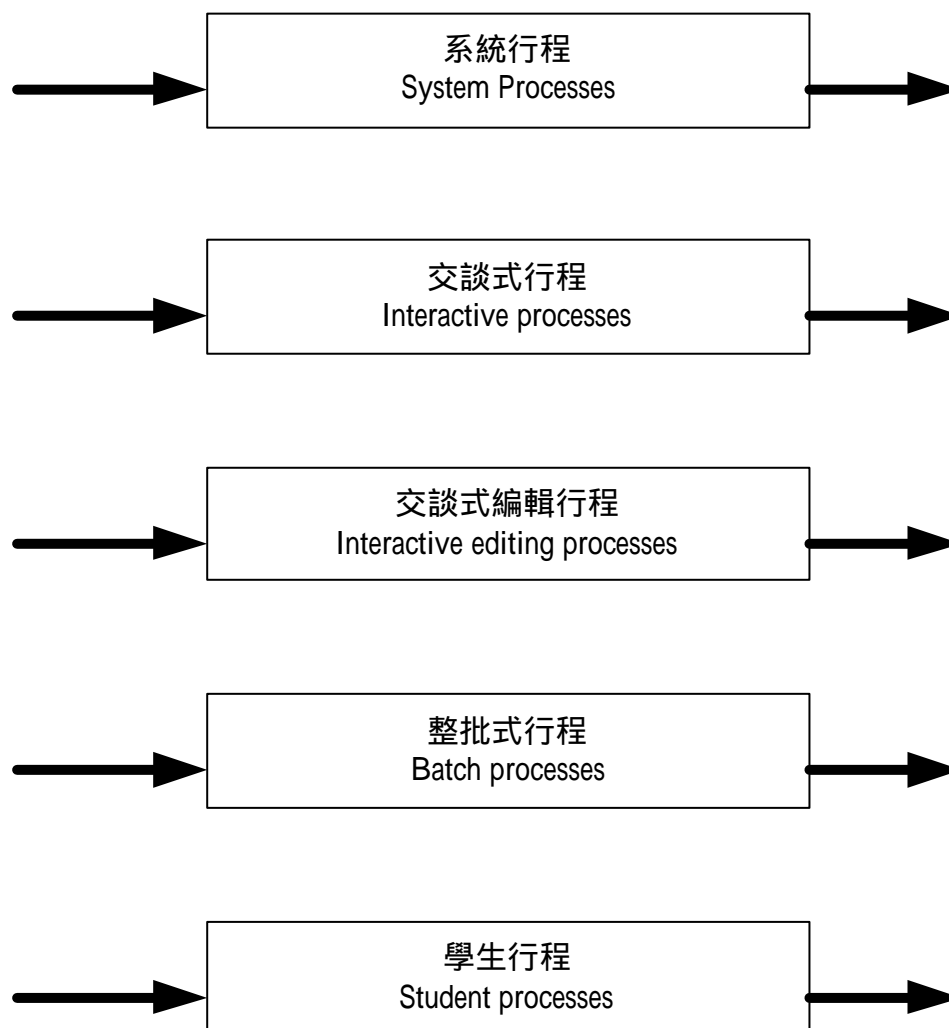


0 3 6 9 12 15 18

這是我們RR的工作排班法，我們假設每次分配CPU給行程3個單位的時間。P2在作完行程前總共等待12秒，P3在作完行程前總共等待15秒，而P1在行程作完前總共等待18秒。因此平均等待時間是 $(12+15+18)/3=15$ 秒。

多層佇列之排班法則 Multilevel Queue Scheduling, 是根據行程的性質將它們分成幾個不同的小組, 最典型的分類方式是區分為 foreground”前景執行”或 background”背景執行”。前景 foreground 代表交談式的行程, 而背景 Background 代表的是整批作業的行程。一般來說前景行程的優先權高於背景執行。多層佇列排班法則將就緒佇列區分為五種獨立的佇列, 行程一般都是按照行程得性質決定, 例如行程的優先權、記憶體的大小、作業系統設計的目的地等。在我們的例子中, 前景的行程使用 Round-Robin 依序循序排班的排班法則, 而背景的行程使用 FCFS 先來先作排班法則。因此, 前景行程的優先權大於常駐行程。FreeBSD 作業系統使用多層佇列之排班法則。

多層佇列之排班法則 Multilevel Queue Scheduling



1-1 Shell 指令的執行過程

當我們在終端機上下達指令，這時預設的 bash(Bourne shell 為父行程)，就會使用 fork 產生子行程，然後子行程會使用 Exec 去執行我們所下達的指令，當指令完成時，而 Bash 父行程又回到 exec 等待我們下達新的指令。

我們在終端機上，下達 ps。

```
#ps
```

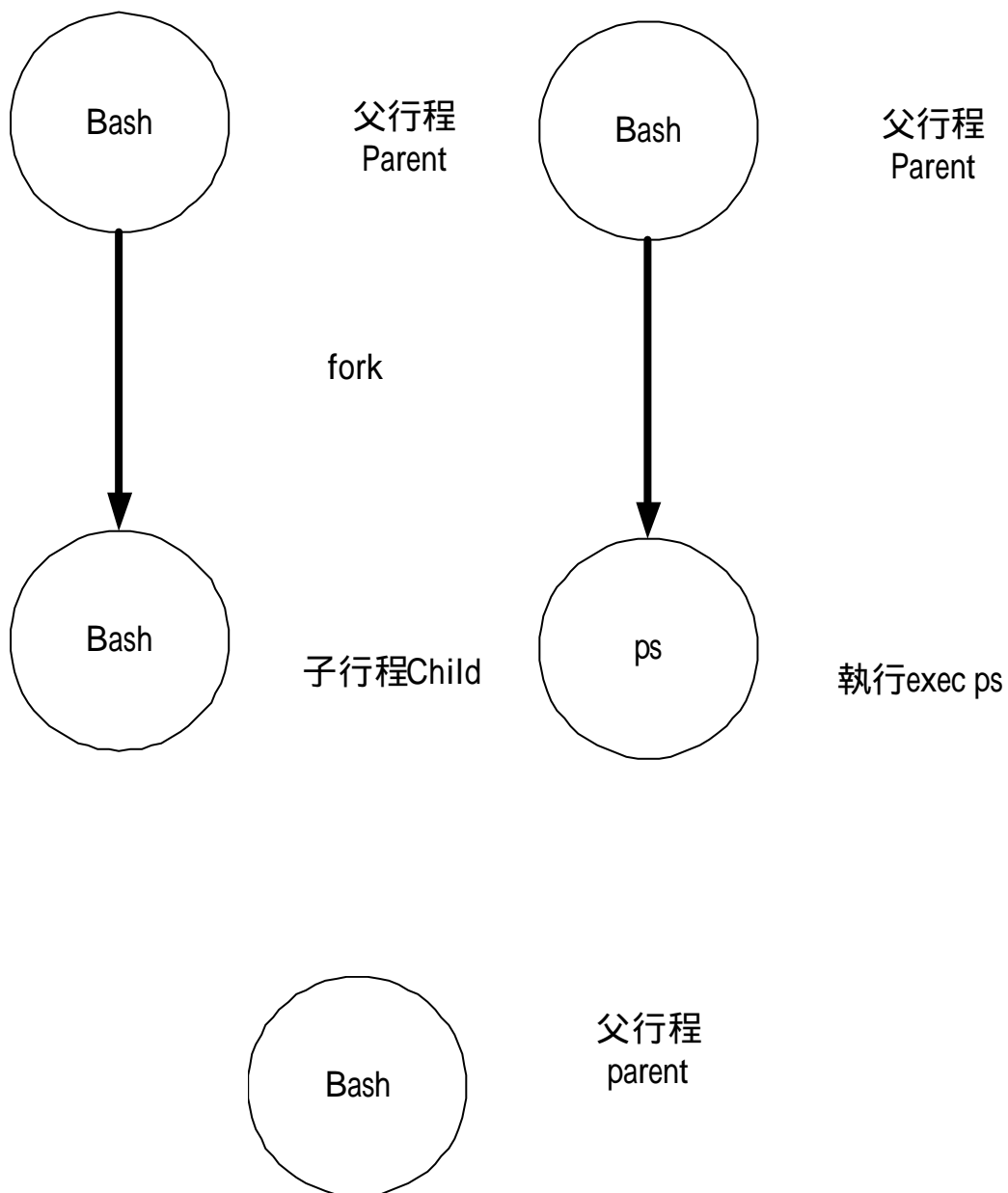
這時 bash 行程 953 就會使用 fork 產生 955 編號的子行程，然後執行 ps。

```
bash-2.05b# ps
  PID  TT  STAT      TIME COMMAND
  655  p0  Is      0:00.03 login [pam] (login)
  658  p0  I       0:00.01 su root
  659  p0  S       0:00.17 _su (csh)
  953  p0  S       0:00.01 bash
  955  p0  R+      0:00.00 ps
  491  v0  Is      0:00.04 login [pam] (login)
  506  v0  I       0:00.02 -csh (csh)
  508  v0  I+      0:00.01 /bin/sh /usr/X11R6/bin/startx
  518  v0  I+      0:00.01 xinit /root/.xinitrc -- -nolisten tcp
  519  v0  S       1:46.45 /usr/X11R6/bin/XFree86 :0 -nolisten tcp
  524  v0  I       0:00.02 /bin/sh /usr/local/bin/startkde
  525  v0  I       0:00.22 xcin2.5
  567  v0  S       0:00.18 kwrapper kmsserver
  492  v1  Is+     0:00.00 /usr/libexec/getty Pc ttyv1
  493  v2  Is+     0:00.00 /usr/libexec/getty Pc ttyv2
  494  v3  Is+     0:00.00 /usr/libexec/getty Pc ttyv3
  495  v4  Is+     0:00.00 /usr/libexec/getty Pc ttyv4
  496  v5  Is+     0:00.00 /usr/libexec/getty Pc ttyv5
  497  v6  Is+     0:00.00 /usr/libexec/getty Pc ttyv6
  498  v7  Is+     0:00.00 /usr/libexec/getty Pc ttyv7
```

執行完 ps 後又回到了指令直譯器。

```
bash-2.05b#
```


當我們在終端機上下達ps指令，這時預設的Bash(Bourne shell為父行程)，就會使用fork產生子行程，然後子行程會使用Exec去執行我們所下達的指令ps，當指令完成時，而Bash父行程又回到exec等待我們下達新的指令。



1-2 行程的屬性

我們可以使用 `ps` 來觀看目前行程的狀態。我們加入一些參數，就可以看到我們所要指定的情況。

指令：`ps`

參數：

-a：顯示所有在終端機執行的行程資訊，包括其它使用者的行程。

-e 或 -A：顯示所有在系統執行的行程資訊。

-j：顯示工作的資訊(包含 Parent PID、group ID,session ID)。

-l：顯示行程的狀態。

-r：顯示狀態正在執行的行程。

-u：顯示使用者行程的資訊。

-x：顯示行程的所有資訊，而沒有包含 TTYs。

-f：顯示行程間的階層與關係。

我們使用 `ps a` 顯示所有在終端機執行的行程資訊，包括其它使用者的行程。

```
bash-2.05b# ps a
  PID  TT  STAT      TIME COMMAND
   655  p0  Is      0:00.03 login [pam] (login)
   656  p0  I        0:00.01 -sh (sh)
   658  p0  I        0:00.01 su root
   659  p0  I        0:00.17 _su (csh)
   953  p0  S        0:00.01 bash
   973  p0  R+       0:00.00 ps a
   491  v0  Is      0:00.04 login [pam] (login)
   506  v0  I        0:00.02 -csh (csh)
   508  v0  I+       0:00.01 /bin/sh /usr/X11R6/bin/startx
   518  v0  I+       0:00.01 xinit /root/.xinitrc -- -nolisten tcp
   519  v0  S        1:46.50 /usr/X11R6/bin/XFree86 :0 -nolisten tcp
   524  v0  I        0:00.02 /bin/sh /usr/local/bin/startkde
   525  v0  I        0:00.22 xcin2.5
   567  v0  S        0:00.19 kwrapper kmsserver
   492  v1  Is+     0:00.00 /usr/libexec/getty Pc ttyv1
   493  v2  Is+     0:00.00 /usr/libexec/getty Pc ttyv2
   494  v3  Is+     0:00.00 /usr/libexec/getty Pc ttyv3
   495  v4  Is+     0:00.00 /usr/libexec/getty Pc ttyv4
   496  v5  Is+     0:00.00 /usr/libexec/getty Pc ttyv5
   497  v6  Is+     0:00.00 /usr/libexec/getty Pc ttyv6
   498  v7  Is+     0:00.00 /usr/libexec/getty Pc ttyv7
```

行程狀態 state	說明
D	非中斷式 sleep(通常為 I/O 輸入/輸出)
N	低優先權行程(被 Nice 過的行程)
R	被排在執行佇列中，隨時會被執行的行程
I	中斷中
S	Sleeping ，正在睡眠中
T	Traced 或 stopped 追蹤或停止
Z	Zombie ，已經被停止的行程
W	被 swapped 到硬碟的行程

行程符號	說明
PID	為行程的編號，每一個行程都有它自己唯一的行程編號。
TTY	行程執行時的終端機。
STAT	行程的狀態
TIME	行程已經執行的時間。
CMD	行程被執行的指令名稱

我們使用 `ps -j` 顯示工作的資訊(包含 Parent PID、group ID,session ID)。

```
bash-2.05b# ps -j
USER  PID  PPID  PGID  JOBC  STAT  TT      TIME  COMMAND
root   655   654   655    0  Is   p0      0:00.03  login [pam] (login)
root   658   656   658    1  I    p0      0:00.01  su root
root   659   658   659    1  I    p0      0:00.17  _su (csh)
root   953   659   953    1  S    p0      0:00.02  bash
root   999   953   999    1  R+   p0      0:00.00  ps -j
root   975   974   975    0  Is+  p1      0:00.03  /bin/csh
root   491    1   491    0  Is   v0      0:00.04  login [pam] (login)
root   506   491   506    1  I    v0      0:00.02  -csh (csh)
```

我們使用 `ps -r` 顯示狀態正在執行的行程。

```
bash-2.05b# ps -r
  PID  TT  STAT      TIME COMMAND
  491  v0  Is      0:00.04 login [pam] (login)
  492  v1  Is+     0:00.00 /usr/libexec/getty Pc ttyv1
  493  v2  Is+     0:00.00 /usr/libexec/getty Pc ttyv2
  494  v3  Is+     0:00.00 /usr/libexec/getty Pc ttyv3
  495  v4  Is+     0:00.00 /usr/libexec/getty Pc ttyv4
  496  v5  Is+     0:00.00 /usr/libexec/getty Pc ttyv5
  497  v6  Is+     0:00.00 /usr/libexec/getty Pc ttyv6
  498  v7  Is+     0:00.00 /usr/libexec/getty Pc ttyv7
  506  v0  I       0:00.02 -csh (csh)
  508  v0  I+      0:00.01 /bin/sh /usr/X11R6/bin/startx
  518  v0  I+      0:00.01 xinit /root/.xinitrc -- -nolisten tcp
  519  v0  S       1:47.75 /usr/X11R6/bin/XFree86 :0 -nolisten tcp
  524  v0  I       0:00.02 /bin/sh /usr/local/bin/startkde
  525  v0  I       0:00.23 xcin2.5
  567  v0  S       0:00.21 kwrapper kmsserver
  655  p0  Is      0:00.03 login [pam] (login)
  658  p0  I       0:00.01 su root
  659  p0  I       0:00.17 _su (csh)
  953  p0  S       0:00.02 bash
  975  p1  Is+     0:00.03 /bin/csh
 1004  p0  R+      0:00.00 ps -r
```

我們使用 `ps -l` 顯示行程的狀態。

```
bash-2.05b# ps -l
  UID  PID  PPID  CPU  PRI  NI   VSZ  RSS  MWCHAN  STAT  TT      TIME COMMAND
    0   655   654    0    8    0  1648 1304  wait    Is    p0      0:00.03 login [pam]
    0   658   656    0    8    0  1616 1220  wait    I     p0      0:00.01 su root
    0   659   658    0   20    0  1536 1184  pause   I     p0      0:00.17 _su (csh)
    0   953   659    0    8    0  1296 1152  wait    S     p0      0:00.03 bash
    0  1006   953    0   96    0   712  544  -       R+    p0      0:00.00 ps -l
    0   975   974    0    5    0  1496 1148  ttyin   Is+   p1      0:00.03 /bin/csh
    0   491    1    0    8    0  1616 1372  wait    Is    v0      0:00.04 login [pam]
    0   506   491    0   20    0  1528 1136  pause   I     v0      0:00.02 -csh (csh)
    0   508   506    0    8    0   896  640  wait    I+    v0      0:00.01 /bin/sh /usr
    0   518   508    0    8    0  2496 1584  wait    I+    v0      0:00.01 xinit /root/
    0   519   518    0   96    0 54076 52808  select  S     v0      1:47.80 /usr/X11R6/b
    0   524   518    0    8    0   916  660  wait    I     v0      0:00.02 /bin/sh /usr
    0   525   524    0   96    0  9896 3524  select  I     v0      0:00.23 xcin2.5
    0   567   524    0    8    0  1276  892  nanslp  S     v0      0:00.21 kwrapper ksm
    0   492    1  133    5    0  1236  912  ttyin   Is+   v1      0:00.00 /usr/libexec
```

我們使用 `ps -u` 顯示使用者行程的資訊。

```
bash-2.05b# ps -u
USER  PID %CPU %MEM    VSZ   RSS  TT  STAT  STARTED      TIME COMMAND
root   491  0.0  0.3  1616  1372  v0  Is    7:12AM    0:00.04 login [pam] (login)
root   492  0.0  0.2  1236   912  v1  Is+   7:12AM    0:00.00 /usr/libexec/getty
root   493  0.0  0.2  1236   912  v2  Is+   7:12AM    0:00.00 /usr/libexec/getty
root   494  0.0  0.2  1236   912  v3  Is+   7:12AM    0:00.00 /usr/libexec/getty
root   495  0.0  0.2  1236   912  v4  Is+   7:12AM    0:00.00 /usr/libexec/getty
root   496  0.0  0.2  1236   912  v5  Is+   7:12AM    0:00.00 /usr/libexec/getty
root   497  0.0  0.2  1236   912  v6  Is+   7:12AM    0:00.00 /usr/libexec/getty
root   498  0.0  0.2  1236   912  v7  Is+   7:12AM    0:00.00 /usr/libexec/getty
root   506  0.0  0.2  1528  1136  v0  I     7:15AM    0:00.02 -csh (csh)
root   508  0.0  0.1   896   640  v0  I+    7:16AM    0:00.01 /bin/sh /usr/X11R6/
root   518  0.0  0.3  2496  1584  v0  I+    7:16AM    0:00.01 xinit /root/.xinitr
root   519  0.0 10.2 54076 52808  v0  S     7:16AM    1:47.84 /usr/X11R6/bin/XFre
root   524  0.0  0.1   916   660  v0  I     7:16AM    0:00.02 /bin/sh /usr/local/
root   525  0.0  0.7  9896  3524  v0  I     7:16AM    0:00.23 xcin2.5
root   567  0.0  0.2  1276   892  v0  S     7:16AM    0:00.21 kwrapper kmsserver
root   655  0.0  0.3  1648  1304  p0  Is    7:38AM    0:00.03 login [pam] (login)
root   658  0.0  0.2  1616  1220  p0  I     7:38AM    0:00.01 su root
root   659  0.0  0.2  1536  1184  p0  I     7:38AM    0:00.17 _su (csh)
root   953  0.0  0.2  1296  1152  p0  S     8:52AM    0:00.03 bash
root   975  0.0  0.2  1496  1148  p1  Is+   8:58AM    0:00.03 /bin/csh
root  1007  0.0  0.1   712   544  p0  R+    9:07AM    0:00.00 ps -u
```

我們使用 `ps -e` 顯示所有在系統執行的行程資訊。

```
bash-2.05b# ps -e
  PID  TT  STAT      TIME COMMAND
  655  p0  Is    0:00.03 login [pam] (login)
  658  p0  I     0:00.01 su root
  659  p0  I     0:00.17 _su (csh)
  953  p0  S     0:00.03 bash
1009  p0  R+    0:00.00 ps -e
  975  p1  Is+   0:00.03 /bin/csh
  491  v0  Is    0:00.04 login [pam] (login)
  506  v0  I     0:00.02 -csh (csh)
  508  v0  I+    0:00.01 /bin/sh /usr/X11R6/bin/startx
  518  v0  I+    0:00.01 xinit /root/.xinitrc -- -nolisten tcp
  519  v0  S     1:47.89 /usr/X11R6/bin/XFree86 :0 -nolisten tcp
  524  v0  I     0:00.02 /bin/sh /usr/local/bin/startkde
  525  v0  I     0:00.23 xcin2.5
  567  v0  S     0:00.21 kwrapper kmsserver
  492  v1  Is+   0:00.00 /usr/libexec/getty Pc ttyv1
  493  v2  Is+   0:00.00 /usr/libexec/getty Pc ttyv2
  494  v3  Is+   0:00.00 /usr/libexec/getty Pc ttyv3
  495  v4  Is+   0:00.00 /usr/libexec/getty Pc ttyv4
  496  v5  Is+   0:00.00 /usr/libexec/getty Pc ttyv5
  497  v6  Is+   0:00.00 /usr/libexec/getty Pc ttyv6
  498  v7  Is+   0:00.00 /usr/libexec/getty Pc ttyv7
```

我們使用 `ps -x` 顯示行程的所有資訊，而沒有包含 TTYs。More 是分段的意義。

```
bash-2.05b# ps -x|more
  PID TT  STAT      TIME COMMAND
    0 ??  DLs    0:00.02 (swapper)
    1 ??  ILs    0:00.01 /sbin/init --
    2 ??  DL     0:00.40 (g_event)
    3 ??  DL     0:01.40 (g_up)
    4 ??  DL     0:04.58 (g_down)
    5 ??  IL     0:00.00 (acpi_task0)
    6 ??  IL     0:00.00 (acpi_task1)
    7 ??  IL     0:00.00 (acpi_task2)
    8 ??  DL     0:00.01 (pagedaemon)
    9 ??  DL     0:00.00 (vmdaemon)
   10 ??  DL     0:00.00 (ktrace)
   11 ??  RL    111:43.23 (idle)
   12 ??  WL     0:00.43 (swil: net)
   13 ??  WL     0:05.47 (swi7: tty:sio clock)
   15 ??  DL     0:00.60 (random)
   24 ??  DL     0:00.00 (usb0)
   25 ??  DL     0:00.00 (usbtask)
   26 ??  WL     0:02.17 (irq14: ata0)
   27 ??  WL     0:00.00 (irq15: atal)
   29 ??  DL     0:00.00 (usb1)
   30 ??  WL     0:00.47 (irq11: rl0)
```

我們使用 `ps -aux` 顯示行程的所有資訊。

```
bash-2.05b# ps -aux|more
USER      PID %CPU %MEM    VSZ   RSS  TT  STAT  STARTED      TIME  COMMAND
root        11  97.2  0.0     0    12  ??  RL    7:12AM 112:43.47 (idle)
root        10  0.0  0.0     0    12  ??  DL    7:12AM 0:00.00 (ktrace)
root         1  0.0  0.1   740  396  ??  ILs   7:12AM 0:00.01 /sbin/init --
root        12  0.0  0.0     0    12  ??  WL    7:12AM 0:00.44 (swil: net)
root        13  0.0  0.0     0    12  ??  WL    7:12AM 0:05.51 (swi7: tty:sio cl
root         2  0.0  0.0     0    12  ??  DL    7:12AM 0:00.40 (g_event)
root         3  0.0  0.0     0    12  ??  DL    7:12AM 0:01.40 (g_up)
root         4  0.0  0.0     0    12  ??  DL    7:12AM 0:04.58 (g_down)
root        15  0.0  0.0     0    12  ??  DL    7:12AM 0:00.60 (random)
root         5  0.0  0.0     0    12  ??  IL    7:12AM 0:00.00 (acpi_task0)
root         6  0.0  0.0     0    12  ??  IL    7:12AM 0:00.00 (acpi_task1)
root         7  0.0  0.0     0    12  ??  IL    7:12AM 0:00.00 (acpi_task2)
root        24  0.0  0.0     0    12  ??  DL    7:12AM 0:00.00 (usb0)
root        25  0.0  0.0     0    12  ??  DL    7:12AM 0:00.00 (usbtask)
root        26  0.0  0.0     0    12  ??  WL    7:12AM 0:02.17 (irq14: ata0)
root        27  0.0  0.0     0    12  ??  WL    7:12AM 0:00.00 (irq15: atal)
root        29  0.0  0.0     0    12  ??  DL    7:12AM 0:00.00 (usb1)
root        30  0.0  0.0     0    12  ??  WL    7:12AM 0:00.47 (irq11: rl0)
root        31  0.0  0.0     0    12  ??  WL    7:12AM 0:00.00 (irq7: ppc0)
root        35  0.0  0.0     0    12  ??  WL    7:12AM 0:00.01 (irq1: atkbd0)
root        36  0.0  0.0     0    12  ??  WL    7:12AM 0:01.72 (irq12: psm0)
```

我們使用 `ps -e f` 就可以看到行程的階層父子關係。

```
[root@flash chaiyen]# ps -e flmore
  PID TTY          STAT       TIME COMMAND
    1 ?           S           0:04  init
    2 ?           SW          0:00  [keventd]
    3 ?           SW          0:00  [kapmd]
    4 ?           SWN        0:00  [ksoftirqd_CPU0]
    5 ?           SW          0:00  [kswapd]
    6 ?           SW          0:00  [bdflush]
    7 ?           SW          0:03  [kupdated]
    8 ?           SW          0:00  [mdrecoveryd]
   12 ?           SW          0:09  [kjournald]
   91 ?           SW          0:00  [khubd]
  622 ?           SW          0:00  [eth0]
```

欄位	說明
PID	為行程的編號，每一個行程都有它自己唯一的行程編號。
TTY	行程執行時的終端機。
STAT	行程的狀態
TIME	行程已經執行的時間。
CMD	行程被執行的指令名稱
USER	行程的執行用者
%CPU	所用 CPU 與所花費的時間的比率
%MEM	記憶體的使用率
VSZ	VIRTUAL SIZE, 行程在記憶體映像中的大小。
RSS	行程在實體記憶體中所佔的大小，單位 KBYTES
START	開使執行行程的時間
F	期標。指出行程是屬於使用者 USER 或核心 Kernel。
UID	行程執行者的使用者 ID
PRI	行程被排班的優先權
PPID	父行程的行程 ID
NI	Nice 的值，Nice 為降低優先權
WCHAN	等待頻道，當為 Null 空時，表示行程正在執行，當行程在就緒時為 Waiting for

當我們要殺掉指定的程式或行程時,我們可以使用 Kill 指令將該行程的編號殺掉及可。我們使用 kill 2643 將編號為 2643 的行程給終止。

語法

指令 : kill

#kill 2643

如果我們想顯示 CPU 即時的資訊,我們可以使用 top 指令。

#top

每格數秒就會更新最新的行程資訊。我們按下 q 就可以離開 top 了。

```
last pid: 1216; load averages: 0.02, 0.01, 0.00 up 0+03:03:59 10:15:30
62 processes: 1 running, 61 sleeping
CPU states: 0.0% user, 0.0% nice, 0.8% system, 0.0% interrupt, 99.2% idle
Mem: 76M Active, 256M Inact, 76M Wired, 632K Cache, 60M Buf, 87M Free
Swap: 1003M Total, 1003M Free
```

PID	USERNAME	PRI	NICE	SIZE	RES	STATE	TIME	WCPU	CPU	COMMAND
550	root	96	0	38868K	22216K	select	1:52	0.05%	0.05%	kdeinit
519	root	96	0	55612K	54344K	select	3:09	0.00%	0.00%	XFree86
576	root	96	0	40532K	25444K	select	0:14	0.00%	0.00%	kdeinit
574	root	96	0	34096K	20340K	select	0:09	0.00%	0.00%	kdeinit
581	root	96	0	32408K	18732K	select	0:08	0.00%	0.00%	kdeinit
562	root	60	-36	7764K	6700K	select	0:08	0.00%	0.00%	artsd
458	root	96	0	1188K	696K	select	0:04	0.00%	0.00%	moused
570	root	96	0	37864K	23424K	select	0:03	0.00%	0.00%	kdeinit
974	root	96	0	36872K	25588K	select	0:03	0.00%	0.00%	kdeinit
572	root	96	0	12208K	9904K	select	0:02	0.00%	0.00%	gimp-1.2
207	root	96	0	3064K	1772K	select	0:01	0.00%	0.00%	ppp
544	root	96	0	19452K	14428K	select	0:01	0.00%	0.00%	kdeinit
946	root	96	0	5300K	3724K	select	0:01	0.00%	0.00%	smbd
654	root	96	0	3272K	2212K	select	0:01	0.00%	0.00%	telnetd
566	root	96	0	25332K	20516K	select	0:01	0.00%	0.00%	kdeinit
441	root	96	0	3704K	2280K	select	0:01	0.00%	0.00%	nmbd

當我們在顯示 CPU 使用情形時，按下 h 就可以看到 top 的功能鍵功能。

A top users display for Unix

These single-character commands are available:

```
^L      - redraw screen
q       - quit
h or ?  - help; show this text
d       - change number of displays to show
e       - list errors generated by last "kill" or "renice" command
i       - toggle the displaying of idle processes
I       - same as 'i'
k       - kill processes; send a signal to a list of processes
n or #  - change number of processes to display
o       - specify sort order (pri, size, res, cpu, time)
r       - renice a process
s       - change number of seconds to delay between updates
u       - display processes for only one user (+ selects all users)
```

當我們按下 u，再輸入使用者，則可以顯示該使用者的情況。

```
last pid: 1216; load averages: 0.00, 0.00, 0.00 up 0+03:05:20 10:16:51
62 processes: 2 running, 60 sleeping
CPU states: 1.0% user, 0.0% nice, 0.9% system, 0.2% interrupt, 98.0% idle
Mem: 76M Active, 256M Inact, 76M Wired, 632K Cache, 60M Buf, 87M Free
Swap: 1003M Total, 1003M Free
```

Username to show:

PID	USERNAME	PRI	NICE	SIZE	RES	STATE	TIME	WCPU	CPU	COMMAND
1202	ju	8	0	1616K	1220K	wait	0:00	0.00%	0.00%	su
658	ju	8	0	1616K	1220K	wait	0:00	0.00%	0.00%	su
656	ju	8	0	900K	644K	wait	0:00	0.00%	0.00%	sh
1200	ju	8	0	900K	644K	wait	0:00	0.00%	0.00%	sh

當我們按下 i 時，就可以顯示暫時閒置 idle 的行程。

```
last pid: 1217; load averages: 0.00, 0.00, 0.00 up 0+03:08:09 10:19:40
62 processes: 1 running, 61 sleeping
CPU states: 0.0% user, 0.0% nice, 1.2% system, 0.0% interrupt, 98.8% idle
Mem: 76M Active, 256M Inact, 76M Wired, 632K Cache, 60M Buf, 87M Free
Swap: 1003M Total, 1003M Free
```

PID	USERNAME	PRI	NICE	SIZE	RES	STATE	TIME	WCPU	CPU	COMMAND
550	root	96	0	38868K	22216K	select	1:54	0.05%	0.05%	kdeinit
1217	root	96	0	2264K	1516K	RUN	0:00	0.00%	0.00%	top