



1-3 行程和工作控制

我們可以在 FreeBSD 上作行程和工作的管理，這包含了新建立(new)行程、行程的終止(zombie)、和正在背景或前景執行的行程、暫停行程、或將行程將背景執行或前景執行作轉換。

1-3-1 前景和背景行程

當我們在終端機上打上指令，然後按下<Enter>，shell 就會執行我們的指令，而且回傳和顯示 shell 提示。當我們的指令在執行時，我們無法再下達新的指令給 shell，直到我們先前的工作已經作完時，而且 shell 回傳時，我們才可以再度下達。當我們的指令以這種方式執行，我們稱為在『前景』foreground 執行。當前景在執行時，它會保留鍵盤和顯示器的控制權。

當有些時後，我們需要在 FreeBSD 上執行一些需要很久時間才能完成的工作，而當這些工作執行時，我們又希望作一些其它的工作。這時我們就需要將指令下到『背景』background 去。當我們下達在『背景』執行的命令時，只要在指令後面加上&(and 的符號)。

背景行程在高 nice 的情況下，也就是低優先權。因此當其它高優先權的行程執行完，才會輪到它使用 CPU。

前景執行

指令：

command

背景執行

指令：

command &

當我們執行 `find` 的指令來尋找和 `mail` 有關的檔案，再將檔案輸入到 `foo.txt` 的檔案中，這是會花費很多的時間。而我們要等待它執行完才能執行下一個動作或指令。

```
[ju@mandrake/]# find / -name mail -print > foo.txt
```

這是我們編輯 `foo.txt` 的情況。

```
#vi foo.txt
```

這是我們使用 `find` 指令找到有關 `mail` 檔案的情況並將它輸入到 `foo.txt`。

```
/usr/bin/mail  
/usr/local/share/emacs/21.3/lisp/mail  
/usr/local/share/apps/ksim/themes/ksim/mail  
/usr/src/etc/mail  
/usr/src/usr.bin/mail  
/usr/ports/chinese/openoffice-zh_TW/work/oo_1.0.3_src/inet/source/mail  
/usr/ports/editors/openoffice/work/oo_1.0.3_src/inet/source/mail  
/usr/ports/mail  
/usr/X11R6/libexec/gimp/1.2/plugin/mail  
/usr/compat/linux/var/mail  
/usr/compat/linux/var/spool/mail  
/var/mail  
/etc/mail
```

當我們執行 `find` 的指令來尋找和 `mail` 有關的檔案並且讓它在背景執行，再將檔案輸入到 `foo.txt` 的檔案中，這是會花費很多的時間，但它是在背景執行。因此我們不需等待它執行完才能下下一個動作或指令。

我們在背景執行 `find` 指令後面加上 `&` 背景符號。而這就是在背景執行的 `find`。

```
bash-2.05b# find / -name mail > -print > foo.txt &
```

有一些工作都需要花費很多的時間來執行，因此放到背景是很好的。例如：sort 排列指令、編譯 cc 或 make 指令、find 尋找檔案指令。我們使用 fg 指令就可以將背景的行程帶到前景來執行。

指令：

fg [%jobid]

參數：

%或%+：目前的工作。

%-：前一個工作。

%N：工作編號 N。

%Name：工作開始的名稱。

%?Name：指令包含名稱。

我們使用 vi 來編輯 foo.txt 檔。

```
#vi foo.txt
```

我們使用 <Ctrl-z> 來暫停使用 vi 編輯 foo.txt 檔。

我們可以使用 jobs 來觀看目前的工作情況，可以看到我們的 vim foo.txt 的狀態是暫停 Stopped。

```
[ju@mandrake/]# jobs
[1]  + Suspended                  vi foo.txt
```

我們現在要再次的編輯它，只要使用 fg %1，就可以將第一個工作給移到前景，這樣我們就可以開始編輯了。

```
[ju@mandrake/]# fg %1
```

我們也可以將被暫停的工作丟到背景去，我們可以使用下列的指令 `bg`。當在編譯 `compile` 檔案時，我們就可以使用 `bg`，將它丟到背景去執行。

指令：

```
bg [%jobid]
```

參數：

`%`或`%+`：目前的工作。

`%-`：前一個工作。

`%N`：工作編號 `N`。

`%Name`：工作開始的名稱。

`%?Name`：指令包含名稱。

當我們要了解有哪一些工作時，可以使用指令 `jobs`。

指令：

```
jobs
```

參數：

`-l`：顯示該工作的行程編號。

我們使用 `jobs -l` 就可以顯示所有的工作，和該工作的行程編號。

```
[ju@mandrake/]# jobs -l
[1] + 1243 Suspended                vi foo.txt
```

`vi` 的行程編號 `PID` 為 1243。

```
[ju@mandrake/]# 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  I+      0:00.17 _su (csh)
  975  p1  Is+     0:00.04 /bin/csh
 1199  p2  Is      0:00.03 login [pam] (login)
 1202  p2  I       0:00.01 su root
 1203  p2  S       0:00.05 _su (csh)
 1243  p2  T       0:00.01 vi foo.txt
 1278  p2  R+      0:00.00 ps
  491  v0  Is      0:00.04 login [pam] (login)
```

1-3-2FreeBSD 常駐行程 Daemons

任何在背景執行的程式都可以叫作 Daemon。Daemons 常駐行程提供各式各樣的服務給我們的使用者，和系統管理的工作，例如：列印、e-mail 都是經過常駐行程的服務。列印服務是由列印常駐行程 Daemon 所提供，而 e-mail 服務是由 smtpd daemon 所提供，而網路瀏覽服務是由 httpd 常駐行程所提供。

1-3-3 指令平行和循序的執行

我們可以在一行命令區輸入多個指令當作是平行或是循序的執行。

這是我們在一行命令區輸入多個指令當作是循序的執行。

語法：

指令 1 ; 指令 2 ; 指令 3 ; ; 指令 N

我們使用分號當作是循序指令的分別。我們第一個是輸入 date 指令，第二個是輸入 echo 指令，我們第三個是輸入 who 指令，它們會循序的先從 date 執行，執行完再執行 echo 指令，最後才會執行 who 指令。

```
[ju@mandrake/]# date;echo "Be Happy";who
Sat Jul 26 10:42:46 CST 2003
Be Happy
root          ttyv0        Jul 26 07:15
root          ttyv2        Jul 26 10:09
ju            ttyp0        Jul 26 07:38 (61-218-29-5.HINE)
root          ttypl        Jul 26 08:58 (mandrake.aasir.c)
ju            ttyv2        Jul 26 10:11 (61-218-29-5.HINE)
```

這是我們在一行命令區輸入多個指令當作是平行的執行(parallel execution)。

語法：

指令 1&指令 2&指令 3&....&指令 N&

我們使用&當作是平行指令的執行。我們第一個是輸入 date 指令，第二個是輸入 echo 指令，我們第三個是輸入 who 指令。它們會平行的執行。我們 date 的行程編號是 1300，然後我們 echo 的行程編號是 1301。

```
[ju@mandrake/]# date & echo "Be Happy" & who
Sat Jul 26 10:44:27 CST 2003
[1] 1300
[2] 1301
Be Happy
root          ttyv0      Jul 26 07:15
root          ttyv2      Jul 26 10:09
ju            ttyp0      Jul 26 07:38 (61-218-29-5.HINE)
root          ttypl      Jul 26 08:58 (mandrake.aasir.c)
ju            ttyv2      Jul 26 10:11 (61-218-29-5.HINE)
[2] - Done          echo Be Happy
[1] + Done          date
```

我們最後也加入&符號，表示 date、echo、who 是平行的執行，因此 date 的行程編號為 1306、ehco 的行程編號為 1307、who 的行程編號為 1308。

```
[ju@mandrake/]# date & echo "Be Happy" & who &
[1] 1306
[2] 1307
Sat Jul 26 10:46:49 CST 2003
Be Happy
root          ttyv0      Jul 26 07:15
root          ttyv2      Jul 26 10:09
ju            ttyp0      Jul 26 07:38 (61-218-29-5.HINE)
root          ttypl      Jul 26 08:58 (mandrake.aasir.c)
ju            ttyv2      Jul 26 10:11 (61-218-29-5.HINE)
[3] 1308
[3] - Done          who
[2] - Done          echo Be Happy
```


1-3-4 指令和行程不正常的中斷

當我們執行指令時，它會執行到結束，最後才中斷。當我們想中斷正在執行的指令，我們可以使用<Ctrl+c>來將前景目前的行程來中斷。我們常用 kill 指令作軟體中斷，並將中斷的信號 siganl 送給行程。Signal 信號可分為內部信號 internal signal 和外部信號 external signal。內部信號 internal signal 是由行程內部自己發出，而外部信號 external signal 則是像<Ctrl+c>是由外部所產生。

1- 4 在 FreeBSD 上的行程階層(Process Hierarchy in FreeBSD)

當我們開啟 FreeBSD 作業系統, LILO(FreeBSD Loader)就會從硬碟載入 FreeBSD 的核心到記憶體。它初始化我們的硬體元件, 像是硬碟控制器(Disk Controller), 然後 FreeBSD 進入保護模式(protected mode)載入作業系統, 然後初始化各種核心資料結構, 像 I-node 和檔案表(file tables)。這個行程的 PID 為 0。它開啟這 init 行程(此行程 PID 為 1), 而 init 行程則執行其它行程的啟動 starts。這個 init 行程啟動 daemons 常駐行程 pagedaemon(分頁常駐行程)、vmdaemon、bufdaemon(緩衝常駐行程)、inetd(網路常駐行程)...。Init 行程然後會初始化檔案系統, 然後掛載到根 root 目錄上, 然後它會執行/sbin/init 的程式, 然後在每個終端機上執行 getty 行程。getty 行程會設定終端機的屬性, 然後它會顯示 login 的畫面, 讓我們登錄系統。當我們登錄系統時, getty 行程就會 fork 子行程, 然後它就會執行 login 的行程這個行程會找尋在/etc/passwd 檔案上的名稱和密碼是否相符。

我們使用 ps -aux|more 顯示行程的 PID 順序。

```
[chaiyen@flash/]#ps -x|more
  PID  TT  STAT      TIME COMMAND
    0  ??  DLs      0:00.01  (swapper)
    1  ??  ILs      0:00.01  /sbin/init --
    2  ??  DL       0:00.33  (g_event)
    3  ??  DL       0:00.26  (g_up)
    4  ??  DL       0:00.35  (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      244:26.51  (idle)
   12  ??  WL       0:00.03  (swi1: net)
   13  ??  WL       0:05.26  (swi7: tty:sio clock)
   15  ??  DL       0:00.23  (random)
   16  ??  WL       0:00.00  (swi6: acpitaskq)
   23  ??  DL       0:00.31  (acpi_thermal)
   24  ??  WL       0:00.09  (irq14: ata0)
   25  ??  WL       0:00.00  (irq15: atal)
   26  ??  DL       0:00.00  (usb0)
   27  ??  DL       0:00.00  (usbtask)
   28  ??  WL       0:00.04  (irq10: r10)
```

```

29 ?? WL 0:00.00 (irq1: atkbd0)
30 ?? WL 0:00.00 (irq12: psm0)
35 ?? WL 0:00.00 (irq7: ppc0)
38 ?? DL 0:01.13 (pagezero)
39 ?? DL 0:00.03 (bufdaemon)
40 ?? DL 0:00.21 (syncer)
41 ?? DL 0:00.03 (vnlru)
42 ?? IL 0:00.00 (nfsiod 0)
43 ?? IL 0:00.00 (nfsiod 1)
44 ?? IL 0:00.00 (nfsiod 2)
45 ?? IL 0:00.00 (nfsiod 3)
247 ?? Ss 0:00.05 /usr/sbin/syslogd -s
336 ?? Ss 0:00.02 /usr/sbin/usbd
383 ?? Is 0:00.19 /usr/sbin/sshd
388 ?? Ss 0:00.29 sendmail: accepting connections (sendmail)
405 ?? Is 0:00.06 /usr/sbin/cron
432 ?? Is 0:00.00 /usr/sbin/moused -p /dev/psm0 -t auto
453 ?? Ss 0:00.01 /usr/sbin/inetd -wW
475 ?? Ss 0:00.59 nmbd
981 ?? Ss 0:00.03 telnetd
982 p0 Ss 0:00.03 login [pam] (login)
984 p0 S 0:00.01 su root
985 p0 D 0:00.02 su (csh)

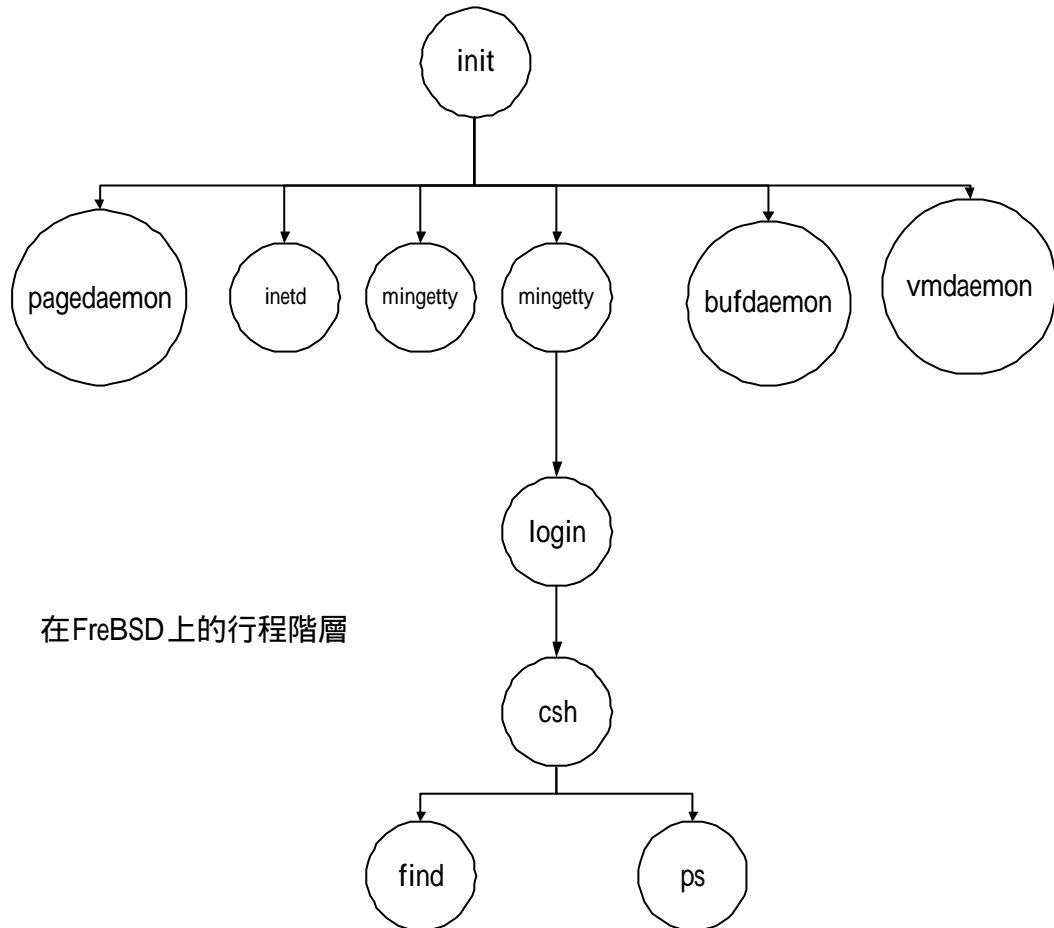
```

```

988 p0 R+ 0:00.00 ps -x
989 p0 RV 0:00.00 _su (csh)
465 v0 Is 0:00.03 login [pam] (login)
473 v0 I+ 0:00.02 -csh (csh)
466 v1 Is+ 0:00.00 /usr/libexec/getty Pc ttyv1
467 v2 Is+ 0:00.00 /usr/libexec/getty Pc ttyv2
468 v3 Is+ 0:00.00 /usr/libexec/getty Pc ttyv3
469 v4 Is+ 0:00.00 /usr/libexec/getty Pc ttyv4
470 v5 Is+ 0:00.00 /usr/libexec/getty Pc ttyv5
471 v6 Is+ 0:00.00 /usr/libexec/getty Pc ttyv6
472 v7 Is+ 0:00.00 /usr/libexec/getty Pc ttyv7

```

在 FreeBSD 上的行程階層，我們可以看到最上層的為 init 行程，而這個行程顯示我們有個使用者正在使用 Bash shell 當作是 login shell 來執行 find 指令和 ps 指令。因此當我們登錄 FreeBSD 時，系統會為我們建立 login 行程，然後建立 login shell，然後這個 shell 執行或直譯我們的指令。



1- 5 開機與關機

當開機時，啟動 FreeBSD 作業系統的順序。

1. BIOS 會執行檢查和測試這臺電腦是否 ok。
2. BIOS 的 bootstrap 載入並啟動為於硬碟的主要啟動記錄(Master Boot Record)。
3. 再來是決定哪一個作業系統要從 MBR(Master Boot Record)被載入。
4. 再來是將/boot/kernel/kernel 核心從/root 檔案系統載入到記憶體中。
5. 核心執行測試和尋找硬體，這時我們就可以在螢幕看到核心找到哪一些硬體，和沒有找到哪一些硬體，這在作業系統啟動過程稱為偵測。
6. 在核心偵測硬體過後，會執行行程編號為 0 的置換記憶體(swapper)，和執行行程編號為 1 的 init(初始化)。置換記憶體 swapper 的工作是在置換主記憶體和虛擬記憶體的空間。Init(初始化)主要的工作是執行啟動系統和長駐行程，當啟動多人使用模式時，它會啟動 shell 來執行/etc/rc 的程式。
7. 一開始/etc/rc 會讀取/etc/defaults/rc.conf 的系統組態設定檔，然後才讀取/etc/rc.conf 的組態設定檔。
8. 當/etc/rc 已經存在時，init(初始化)就會讀取/etc/ttys 和啟動行程。

我們使用 shutdown 或 halt 指令就可以關機。這是指定在早上 8:00 關機。

```
[root@flash rc.d]# shutdown -h 8:00
```

這是指定電腦在三分鐘後關機。

```
[root@flash rc.d]# shutdown -h +3
```

這是指定電腦立刻關機。

```
[root@flash rc.d]# halt
```

這是指定電腦重新開機。

```
[root@flash rc.d]# reboot
```

將電腦關機。

語法：

指令：shutdown

將電腦關機。

語法：

指令：halt

將電腦重新開機。

語法：

指令：reboot

1-5-1cron 定時執行程式

/etc/periodic/daily 是系統每天 am2:00 就會執行的程式，/etc/periodic/weekly 是每個禮拜六 am3:30 就會執行的程式，/etc/periodic/monthly 是每個月第一天的 am5:30 就會執行的程式，這些都是 FreeBSD 在固定的時間就會執行的程式。

要告訴 cron 要在特定時間執行程式，我們需要修改/etc/crontab。

#是表示註解。

每一行有七個欄位，minute 分、hour 時、mday 每個月的第幾天、month 月、wday 禮拜幾、擁有者、指令。

*代表每個。例如 0 2 * * *代表 0 分、2 點(以 24 小時)、每天、每個月、每個禮拜。

下列表示每隔三分鐘，root 超級使用者就會執行 find 指令，去尋找檔名為 good.c 的檔案。

```
#/3 * * * * * root find / -name good.c
```

```
# /etc/crontab - root's crontab for FreeBSD
#
# $FreeBSD: src/etc/crontab,v 1.32 2002/11/22 16:13:39 tom Exp $
#
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
HOME=/var/log
#
#minute hour    mday    month    wday    who    command
#
*/5 * * * * * root    /usr/libexec/atrun
*/3 * * * * * root    find / -name good.c
#
# Save some entropy so that /dev/random can re-seed on boot.
*/11 * * * * * operator /usr/libexec/save-entropy
#
# Rotate log files every hour, if necessary.
0 * * * * * root    newsyslog
#
# Perform daily/weekly/monthly maintenance.
1 3 * * * * root    periodic daily
15 4 * * * 6 root    periodic weekly
30 5 1 * * * * root    periodic monthly
```

這是還未由 cron 自動啟動 find 指令。

```
#ps -x
```

```
462 ?? Is      0:00.01 /usr/sbin/moused -p /dev/psm0 -t auto
483 ?? Is      0:00.14 /usr/sbin/inetd -wW
799 ?? Ss      0:00.05 telnetd
811 ?? Ss      0:00.03 telnetd
800 p0 Is      0:00.03 login [pam] (login)
803 p0 I       0:00.01 su root
804 p0 S+     0:00.03 _su (csh)
812 p1 Is      0:00.03 login [pam] (login)
815 p1 I       0:00.01 su root
816 p1 S       0:00.02 _su (csh)
835 p1 R+     0:00.00 ps -x
495 v0 Is      0:00.03 login [pam] (login)
503 v0 I+     0:00.04 -csh (csh)
496 v1 Is+    0:00.00 /usr/libexec/getty Pc ttyv1
497 v2 Is+    0:00.00 /usr/libexec/getty Pc ttyv2
498 v3 Is+    0:00.00 /usr/libexec/getty Pc ttyv3
499 v4 Is+    0:00.00 /usr/libexec/getty Pc ttyv4
500 v5 Is+    0:00.00 /usr/libexec/getty Pc ttyv5
501 v6 Is+    0:00.00 /usr/libexec/getty Pc ttyv6
502 v7 Is+    0:00.01 /usr/libexec/getty Pc ttyv7
```

過了三分鐘，我們看到行程編號 836 號的 cron 行程，啟動了行程編號 837 的 find 指令。

```
#ps -x
```

```
462 ?? Is      0:00.01 /usr/sbin/moused -p /dev/psm0 -t auto
483 ?? Is      0:00.14 /usr/sbin/inetd -wW
799 ?? Ss      0:00.05 telnetd
811 ?? Ss      0:00.04 telnetd
836 ?? I       0:00.00 cron: running job (cron)
837 ?? Is      0:00.00 /bin/sh -c find / -name good.c
838 ?? D       0:02.19 find / -name good.c
800 p0 Is      0:00.03 login [pam] (login)
803 p0 I       0:00.01 su root
804 p0 S       0:00.03 _su (csh)
839 p0 R+     0:00.00 ps -x
812 p1 Is      0:00.03 login [pam] (login)
815 p1 I       0:00.01 su root
816 p1 S+     0:00.02 _su (csh)
495 v0 Is      0:00.03 login [pam] (login)
503 v0 I+     0:00.04 -csh (csh)
496 v1 Is+    0:00.00 /usr/libexec/getty Pc ttyv1
497 v2 Is+    0:00.00 /usr/libexec/getty Pc ttyv2
498 v3 Is+    0:00.00 /usr/libexec/getty Pc ttyv3
499 v4 Is+    0:00.00 /usr/libexec/getty Pc ttyv4
500 v5 Is+    0:00.00 /usr/libexec/getty Pc ttyv5
501 v6 Is+    0:00.00 /usr/libexec/getty Pc ttyv6
502 v7 Is+    0:00.01 /usr/libexec/getty Pc ttyv7
```

1- 6 使用 Webmin 管理行程

我們可以輕易的透過網路使用 Webmin，來遙控遠端電腦程式的執行，這就是 Webmin 管理行程。我們選取系統，就可以看到和行程有關的項目，有執行中的程序、指令排程、定時執行工作等。我們選取系統。

顯示執行中的行程。執行中的程序就是我們 top 指令所列出的程序。



可以設定預備要執行的行程。



我們經常有定時性的工作，例如每日零晨開始發送 E-Mail 或自動執行一些備份的工作，這時我們就需要定時執行工作(Cron)。



1-6-1 執行中的程序

我們選取系統再選取執行中的行程，我們就可以看到執行中的程序。我們可以輸入一個要執行的命令的指令。因為標準的系統命令殼 `/bin/sh` 會被用於執行這個命令，所以我們可以使用一些命令殼的操作子。

執行中的程序

顯示方式： 依據 PID (程序樹) 依據使用者 依據記憶體使用量 依據 CPU 使用量 搜尋 執行...

要執行的命令 執行

執行模式 在背景中執行 等待直到執行完畢

給命令的輸入

我們輸入 `ls` 就可以顯示目錄的檔案。

要執行的命令 執行

我們可以選取依據 PID 來排列執行中的行程。

執行中的程序

顯示方式： 依據 PID (程序樹) 依據使用者 依據記憶體使用量 依據 CPU 使用量 搜尋 執行...

程序編號	擁有者	命令
<u>1</u>	root	init
2	root	[keventd]
<u>3</u>	root	[kapmd]
4	root	[ksoftirqd_CPU0]
<u>5</u>	root	[kswapd]
<u>6</u>	root	[bdflush]
<u>7</u>	root	[kupdated]
<u>8</u>	root	[mdrecoveryd]
82	root	[khubd]
206	root	[kjournald]
600	root	[eth0]
678	root	syslogd -m 0
683	root	klogd -x
703	rpc	portmap
731	rpcuser	rpc.statd
859	root	/usr/sbin/apmd -p 10 -w 5 -W -P /etc/sysconfig/apm-scripts/apmscri
<u>929</u>	ident	identd

我們也可以使用指定使用者的方式來看該使用者正在執行的行程，這是超級使用者 root 的行程執行。

root (root)

程序編號	CPU 使用量	命令
3405	22.0 %	/usr/local/webmin-0.990/proc/index_user.cgi
1	0.1 %	init
4	0.0 %	[ksoftirqd_CPU0]
5	0.0 %	[kswapd]
6	0.0 %	[bdflush]
7	0.0 %	[kupdated]
8	0.0 %	[mdrecoveryd]
82	0.0 %	[khubd]
206	0.0 %	[kjournald]
600	0.0 %	[eth0]

我們也可以依據記憶體使用的情況來排列正在執行的行程。

執行中的程序

顯示方式： 依據 PID (程序樹) 依據使用者 依據記憶體使用量 依據 CPU 使用量 搜尋 執行...

Real memory: 514300 kB total / 309480 kB free Swap space: 1228964 kB total / 1228964 kB free

程序編號	擁有者	記憶體用量	命令
1321	apache	81416 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1322	apache	81416 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1323	apache	81416 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1324	apache	81416 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1325	apache	81416 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1326	apache	81416 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1327	apache	81416 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1328	apache	81416 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1302	root	81368 kB	/usr/sbin/httpd -DHAVE_ACCESS -DHAVE_PROXY -DHAVE_AUTH_ANON ...
1664	root	42844 kB	/usr/bin/X11/X :0 -auth /var/gdm/:0.Xauth
1258	mysql	27920 kB	/usr/libexec/mysqld --defaults-file=/etc/my.cnf --basedir=/u ...
1244	mysql	27920 kB	/usr/libexec/mysqld --defaults-file=/etc/my.cnf --basedir=/u ...
1257	mysql	27920 kB	/usr/libexec/mysqld --defaults-file=/etc/my.cnf --basedir=/u ...
1264	mysql	27920 kB	/usr/libexec/mysqld --defaults-file=/etc/my.cnf --basedir=/u ...

我們也可以依據 CPU 的使用量來排列正在執行的行程。

執行中的程序

顯示方式： [依據 PID \(程序樹\)](#) [依據使用者](#) [依據記憶體使用量](#) **依據 CPU 使用量** [搜尋](#) [執行...](#)

CPU load averages: 0.01 (1 mins) , 0.00 (5 mins) , 0.00 (15 mins)

程序編號	擁有者	CPU 使用量	命令
<u>1</u>	root	0.1 %	init
<u>1371</u>	squid	0.1 %	(squid) -D
<u>3</u>	root	0.0 %	[kapmd]
<u>4</u>	root	0.0 %	[ksoftirqd_CPU0]
<u>5</u>	root	0.0 %	[kswapd]
<u>6</u>	root	0.0 %	[bdflush]
<u>7</u>	root	0.0 %	[kupdated]
<u>8</u>	root	0.0 %	[mdrecoveryd]
<u>82</u>	root	0.0 %	[khubd]
<u>206</u>	root	0.0 %	[kjournald]
<u>600</u>	root	0.0 %	[eth0]
<u>678</u>	root	0.0 %	syslogd -m 0
<u>683</u>	root	0.0 %	klogd -x

這個表單可以讓我們以特定的條件搜尋在系統上的程序。當按下搜尋按鈕號，一組符合條件的列表將會被顯示在下面。對於每一個符合的條件，都會顯示出程序編號、擁有者、CPU 使用量與實際執行的命令。按下程序編號可以顯示更多的程序相關訊息。

在表單下面的按鈕可以讓我們送出控制訊號給符合搜尋條件的程序。請從列表中選取您需要送出的控制訊號，並且按下送出控制訊號按鈕。最常用的控制訊號為 KILL 與 TERM，用以刪除這個程序。我們可以使用搜尋來搜尋特定正在執行的行程。我們在 port 埠輸入 3128 來找尋相關在 3128 埠執行的行程，結果我們找到 squid 行程。

執行中的程序

顯示方式： [依據 PID \(程序樹\)](#) [依據使用者](#) [依據記憶體使用量](#) [依據 CPU 使用量](#) **搜尋** [執行...](#)

擁有者 ...

符合條件

Using more than % CPU

使用檔案系統

使用檔案 ...

Using port protocol

搜尋 Ignore search processes in result

程序編號	擁有者	CPU 使用量	命令
<u>1368</u>	squid	0.0 %	(squid) -D

送出控制信號

1-6-2 指令排程

我們可以設定預備要執行的行程。我們輸入以 goddess 使用者來執行預備要執行的工作。

指令排程

工作識別碼	使用者	執行時間	建立時間	執行指令
1	goddess	Fri Aug 23 05:55:00 2002	Fri Aug 23 05:50:45 2002	ls
2	goddess	Fri Aug 23 05:52:00 2002	Fri Aug 23 05:51:14 2002	ps -s

新指令排程

使用者:

日期: 23 / 八月 / 2002 時間: :

今天日期: 23/八月/2002 目前時間: 05:51

在哪目錄執行:

執行指令:

1-6-3 定時執行工作

我們經常有定時性的工作，例如每日零晨開始發送 E-Mail 或自動執行一些備份的工作，這時我們就需要定時執行工作(Cron)。我們可以選取建立新的定時執行工作。

定時執行工作 (Cron)

建立新的定時執行工作

使用者	是否啓動?	命令
root	是	/etc/cron.hourly/inn-cron-nntpsend /etc/cron.hourly/inn-cron-rnews /etc/cron.hourly/diskcheck
	是	/etc/cron.daily/makewhatis.cron /etc/cron.daily/logrotate /etc/cron.daily/0anacron /etc/cron.daily/rpm /etc/cron.daily/slocate.cron /etc/cron.daily/tmpwatch /etc/cron.daily/inn-cron-expire /etc/cron.daily/00-logwatch /etc/cron.daily/tripwire-check /etc/cron.daily/00webalizer /etc/cron.daily/slrnpull-expire /etc/cron.daily/tetex.cron

我們可以設定建立定時執行工作，我們可以設定定時執行的工作者、命令，然後我們在設定命令執行的時間。

建立定時執行工作

工作詳細資料

執行定時執行工作的使用者 ... 是否啓動？ 是 否

命令

給命令的輸入

執行的時間

分	時	日	月	星期
<input type="radio"/> 全部 <input type="radio"/> 選擇的...	<input type="radio"/> 全部 <input type="radio"/> 選擇的...	<input type="radio"/> 全部 <input type="radio"/> 選擇的...	<input type="radio"/> 全部 <input type="radio"/> 選擇的...	<input type="radio"/> 全部 <input type="radio"/> 選擇的...
0 12 24 36 48	0 12	1 13 25	一月	星期日
1 13 25 37 49	1 13	2 14 26	二月	星期一
2 14 26 38 50	2 14	3 15 27	三月	星期二
3 15 27 39 51	3 15	4 16 28	四月	星期三
4 16 28 40 52	4 16	5 17 29	五月	星期四
5 17 29 41 53	5 17	6 18 30	六月	星期五
6 18 30 42 54	6 18	7 19 31	七月	星期六
7 19 31 43 55	7 19	8 20	八月	
8 20 32 44 56	8 20	9 21	九月	
9 21 33 45 57	9 21	10 22	十月	
10 22 34 46 58	10 22	11 23	十一月	
11 23 35 47 59	11 23	12 24	十二月	

1-6-4 開機關機與行程管理

我們可以使用 Webmin 系統的開機關機選項來啟動或關閉行程。



這是每個行程動作的情形。

開機與關機

建立新的開機或關機動作

動作	Start at boot?	說明
<input type="checkbox"/> aep1000	否	load and unload AEP1000/AEP2000 coprocessor driver
<input type="checkbox"/> amd	是	Runs the automount daemon that mounts devices and NFS hosts on demand.
<input type="checkbox"/> anacron	是	Run cron jobs that were left out due to downtime
<input type="checkbox"/> apmd	是	apmd is used for monitoring battery status and logging it via syslog(8). It can also be used for shutting down the machine when the battery is low.
<input type="checkbox"/> arpwatch	否	The arpwatch daemon attempts to keep track of ethernet/ip address pairings.
<input type="checkbox"/> atalk	否	This package enables Linux to talk to Macintosh computers via the AppleTalk networking protocol. It includes a daemon to allow Linux to act as a file server over EtherTalk or IP for Mac's.
<input type="checkbox"/> atd	是	Runs commands scheduled by the at command at the time specified when at was run, and runs batch commands when the load average is low enough.

當我們選取重新啟動系統就可以重新啟動系統，當我們選取關閉系統就會關機。

重新啟動系統	按下這個按鈕以立刻重新啟動這個系統。所有目前登入的使用者都會斷線，而且所有的服務都會重新啟動。
關閉系統	按下這個按鈕以立刻關閉這個系統。所有的服務都會停止，所有的使用者都會斷線而且系統的電源會關閉（如果您的硬體支援的話）。

我們可以選取 smb(samba 伺服器)則我們可以重新啟動或關閉它。

<input type="checkbox"/> sendmail	是	Sendmail is a Mail Transport Agent, which is the program that moves mail from one machine to another.
<input type="checkbox"/> single	否	
<input type="checkbox"/> smartd	否	Self Monitoring and Reporting Technology (SMART) Daemon
<input type="checkbox"/> smb	是	Starts and stops the Samba smbd and nmbd daemons used to provide SMB network services.

這是 smb(samba)的編輯動作。

編輯動作

動作詳細資料

名稱: smb

動作命令稿:

```
#!/bin/sh
#
# chkconfig: - 91 35
# description: Starts and stops the Samba smbd and nmbd daemons \
#               used to provide SMB network services.
#
# pidfile: /var/cache/samba/smbd.pid
# pidfile: /var/cache/samba/nmbd.pid
# config: /etc/samba/smb.conf

# Source function library.
if [ -f /etc/init.d/functions ]; then
    . /etc/init.d/functions
elif [ -f /etc/rc.d/init.d/functions ]; then
```

是否在開機時啟動? 是 否 Started now? 是

儲存 立刻啟動 Restart Now Restart If Needed Reload Now Show Status 立刻停止

我們選取 show states 就可以顯示 samba 的狀態。

Action Status

執行 /etc/rc.d/init.d/smb status 中...

```
smbd (pid 1555) is running...
nmbd (pid 1560) is running...
```

我們也可以重新啟動 smb。

```
Restart Action

執行 /etc/rc.d/init.d/smb restart 中...

Shutting down SMB services: [ OK ]
Shutting down NMB services: [ OK ]
Starting SMB services: [ OK ]
Starting NMB services: [ OK ]
```

1-6-5 SysV 系統啟動組態

我們可以使用 Webmin 系統的 SysV 系統啟動組態來設定啟動時的系統。



SysV 系統啟動就是 /etc/inittab 啟動。我們可以看到名稱 id 為 5(啟動時開啟 X 視窗)。si 為系統初使化檔。

SysV 系統啟動組態			
建立一個新的啟動程序			
名稱	執行層級	動作方式	啟動程序
id	5	系統開機後	無
si	無	開機程序中	/etc/rc.d/rc.sysinit
l0	無	等待	/etc/rc.d/rc 0
l1	1	等待	/etc/rc.d/rc 1
l2	2	等待	/etc/rc.d/rc 2
l3	3	等待	/etc/rc.d/rc 3
l4	4	等待	/etc/rc.d/rc 4
l5	5	等待	/etc/rc.d/rc 5
l6	6	等待	/etc/rc.d/rc 6
ud	無	只執行一次	/sbin/update
ca	無	暖開機 (Ctrl-Alt-Del)	/sbin/shutdown -t3 -r now
pf	無	失去電源時, 不等待	/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr	1, 2, 3, 4, 5	電源回復時	/sbin/shutdown -c "Power Restored; Shutdown Cancelled"