第七章 行程與服務管理

7-1 行程簡介

7-1-1 行程與程式

『行程』是系統最小的程式執行單元，並享有獨立的記憶體空間。當行程被啟動之後，它便參與 CPU 排程並準備執行。行程一旦被 CPU 選入執行，它除了具有原 CPU 容許最大的記憶體空間之外，對於整個主機系統的資源設備也享有所有主控權（依照行程的權限而定）。簡單的說，在行程被選入執行的期間，所有主機電腦的資源都歸它掌控，但所能管理的程度仍需視該行程的權限而定。

『行程』（Process）與『程式』（Program）之間的差別，一直以來不斷困擾著初學者。簡單的說，程式是靜態還未執行的工作範本；而行程是依照工作範本所執行中的程式，它具有生命週期並進行活動中。此外，同一種工作範本（程式），可同時產生並進行多個執行工作，如圖 7-1 所示。

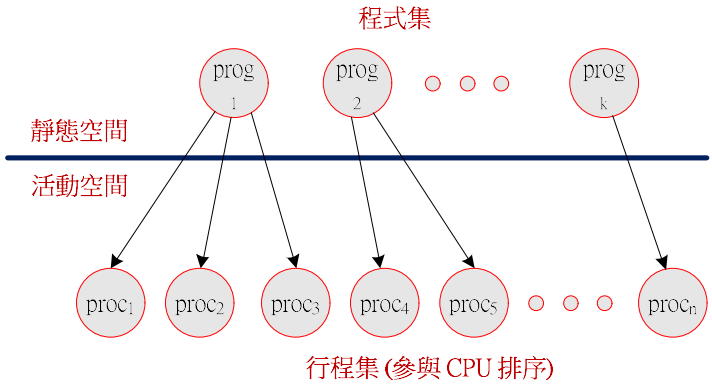

圖 7-1 行程與程式

基本上，行程必須由另一個行程所產生，產生行程者稱之為『父行程』（Parent process），被產生者則稱為『子行程』（Child process）。當子行程被產生時（fork() 系統呼叫），它會繼

承父行程的工作環境，但還未真正被啟動；接著父行程再將程式嵌入子行程並啟動它，如圖 7-2 所示。另一個重點是，當父行程因某種原因被中止之後，所有由它所產生的子行程也將隨之被刪除。為了方便管理，每一個執行當中的行程都有一個序號，稱為『行程識別碼』（Process Identifier, PID）。

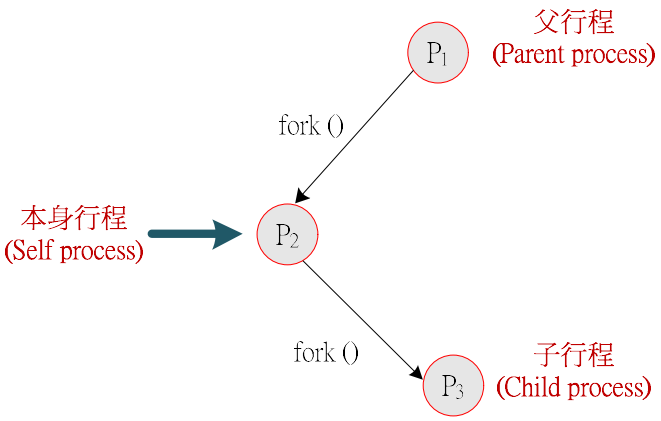

圖 7-2 行程的產生

既然行程是由它的父行程所產生，父行程必然也是由它的父行程所產生，所以系統必然存在一個始作俑者，它就是 init（PID=1）程序。也就是說，當系統啟動時第一個產生的行程是 init，其他的行程皆由它來產生。

7-1-2 行程概述

我們將行程的重點歸納如下：

- 系統中第一個行程為 init（PID=1）。
- 每一個行程都有一個編號，稱為 PID（Process ID）。
- 執行中的行程，呼叫 fork() 系統呼叫可以產生另一個行程。行程產生另一個行程，新的行程稱為原行程的『子行程』(child process)，而產生子行程者稱之為『父行程』(parent process)。
- 當父行程被消滅時，所有由它所產生的子行程也必須隨之消失，否則稱之為『孤兒』（Orphan）。

- 每一行程都有一個獨立的運作環境，行程之間也可透過其他機制來互相通訊，而此獨立的運作環境需要一塊記憶體容量來儲存。
- 至於一部主機可容納多少行程產生，這與該主機 CPU 的處理能力有關，並且也與記憶體的容量息息相關。

照理說，當父行程停止後，子行程便沒有存在的價值，應該隨之被消滅才對，但系統還是有可能發生異常狀況，讓某些子行程被遺留下來。因此，一般系統都會在某一區段時間內搜尋是否有『孤兒』存在，如發現有再將其刪除，否則它會繼續參與 CPU 行程排班，並可能佔用其他系統資源，造成系統執行一些沒有意義的工作。

7-2 行程管理命令

了解行程的概念之後，接下來介紹幾種行程的管理命令。

7-2-1 行程操作命令 - ps

(A) 命令格式

使用者可利用 ps (Process status) 命令來觀察目前系統有哪些行程正在執行，也可藉由 ps 來了解執行中行程的狀態如何，命令格式如下：

```
$ ps [options]
```

常用選項有：

- pids：顯示指定 PID 的行程。
- -e ：顯示所有行程。
- -l ：長行顯示格式。
- -f ：顯示所有資訊。
- -u ：顯示某一使用者所產生的行程。

- --sort：排序顯示。

以下藉由幾種操作範例來說明 ps 命令的使用方法，說明如下：

(B) 顯示自己的行程

使用者直接輸入 ps 命令後，可觀察到自己所下的行程如何，範例如下：

```
$ ps
  PID TTY          TIME CMD
30086 pts/0    00:00:00 bash
30145 pts/0    00:00:00 ps
```

其中：

- PID：行程識別碼，屬隨機產生。
- TTY：該行程所產生的終端機，pts/0 表示第 0 號虛擬終端機，也就是網路終端機。
- TIME：該行程已執行了多少時間。
- CMD：產生該行程的命令，也就是 Shell 的命令名稱。

(C) 顯示系統的行程

使用者如欲觀察整個系統目前有哪些行程正在執行，可增加其選項為 ps –ef，範例如下：

```
$ ps   -ef
ID          PID   PPID   C STIME TTY          TIME CMD
root          1      0   0 Nov04 ?        00:00:03 init
root          2      1   0 Nov04 ?        00:00:00 [keventd]
....
root      30085  30084   0 14:46 ?        00:00:00 [login]
tsnien    30086  30085   0 14:46 pts/0    00:00:00 -bash
tsnien    30225  30086   0 17:24 pts/0    00:00:00 ps -ef
```

其中：

- ID：產生此行程的使用者 ID。
- PPID：產生此行程的父行程 ID（Parent Process ID）。
- STIME：產生此行程的日期時間。

7-2-2 行程的關聯 - pstree

我們可利用 pstree 命令，以樹狀結構來觀察行程之間的關連（子行程與父行程），命令格式如下：

```
$ pstree    [options]
```

常用選項有：

- -a：顯示每個行程完整路徑。
- -h ：在樹狀結構中標示目前執行的行程。
- -l ：長行顯示格式。
- -n ：使用行程識別碼排序。
- -u ：顯示使用者名稱。

操作範例如下：

```
[tsnien@Secure-1 ~]$ pstree    -u
init─┬─acpid
     ├─atd
     ├─automount
     ├─avahi-daemon(avahi)───avahi-daemon
     ├─bluez-pin
     ├─bonobo-activati
     ├─clock-applet
     ├─crond
```

```
        ├─cups-config-dae
        ├─cupsd
        ├─dbus-daemon───{dbus-daemon}
        ├─dbus-daemon(dbus)───{dbus-daemon}
        ├─dbus-launch
        ├─eggcups
……
```

7-2-3 行程的中止 – kill

執行中的行程也可以被強迫中止，但一般使用者僅能刪除自己所產生的行程，不可以中止其他行程。而系統管理員(root 使用者)則享有最高權限，可中止任何行程，其中包含 init (PID=1) 行程。中止行程的命令為 kill，格式如下：

```
# kill [ -s signal | -p ] [ -n ] [ -- ] pid ...
```

常用選項有：

- **-n** ：n 為大於 0 的整數，最高為 9，表示強制性的高低。
- **pid** ：行程的 PID。

選項 n 表示強制性的多寡。行程在執行當中可能正在處理某一特定的工作，而此工作可能具有某些重要性的任務，它是否可以及時被刪除也可能需要思考一下。如果選項為 -9，則無論行程目前執行的工作有何等重要，都要及時被刪除，至於 -0 則需等待一段時間之後，才可以刪除該行程。中止行程必須指名欲被刪除行程的識別碼 (PID)，如果不知道的話，可利用 ps –ef 命令來查詢所欲刪除行程的 PID 號碼；強制刪除背景程式 vi (vim) 的操作範例如下：

```
$ vi test.c &                                    【產生一個背景行程】
[1] 19553
$ ps                                             【顯示自己的行程】
```

```
  PID TTY          TIME CMD
19341 pts/85     00:00:00 bash
19553 pts/85     00:00:00 vim
$ kill   -9   19553                                    【刪除某一行程】
$ ps                                                   【顯示刪除結果】
  PID TTY          TIME CMD
19341 pts/85     00:00:00 bash
19555 pts/85     00:00:00 ps
[1]+   Killed                  vim test.c
```

7-3 行程記錄目錄

7-3-1 系統行程目錄 - /proc

系統在正常運作情況下，會產生許多動態的行程，以及系統運作的相關參數，都會被儲存於 /proc/ 目錄底下；基本上，這些訊息的內容都是隨時變動的，並保存於主記憶體內，但為了系統操作將他們以磁碟檔案格式儲存。再掛載於 /proc/ 目錄底下，我們首先觀察 /proc/ 目錄下有哪些檔案（本書範例）：

```
[root@Linux-1 ~]# ls   /proc
1      1670  2074  2226  2605  2722  4270        buddyinfo     irq           self
1237   1685  2083  223   2609  2724  4272        bus           kallsyms
slabinfo
1238   1687  2113  2231  2634  2737  4296        cmdline       kcore         stat
1239   1710  2123  2243  2637  2739  4298        cpuinfo       keys
swaps
1240   1831  2132  2265  2638  2741  4319        crypto        key-users     sys
.............
1594   2024  2168  2483  2707  2818  87          fs            mounts
1611   2033  2175  2488  2709  3     925         ide           mtrr
1629   2042  2190  2578  2711  388   acpi        interrupts    net
1643   2059  2206  2602  2713  4     asound      iomem         partitions
1647   2065  2217  2603  2716  4269  bluetooth   ioports       pci
```

上述範例中，每一個檔案或目錄表示一個行程或核心參數記錄，其中數字是行程的 PID（Process ID）號碼；以目錄名稱 4269 為例，它代表著 PID=2693 的行程（本書範例，sshd），可由此目錄下觀察出該行程的狀態，操作範例如下：

```
$ ps -ef | grep   sshd          [查閱 sshd 行程]
root       1487      1   0 10:20 ?          00:00:00 /usr/sbin/sshd
```

```
root        2676   1487   0 10:20 ?        00:00:00 sshd: student01 [priv]
student+    2693   2676   0 10:21 ?        00:00:00 sshd: student01@pts/0
student+    2786   2696   0 10:25 pts/0    00:00:00 grep --color=auto sshd
$ cat /proc/2693/status                  [查閱行程狀態]
Name:    sshd
State:   S (sleeping)
.....
```

7-3-2 使用者登入範例

系統啟動後，第一個程序為 /sbin/init，它會根據 RC Scrip 設定檔，啟動執行系統所需的行程。可利用 ps –ef 命令觀察，如下：

```
# ps -ef | grep init
root         1      0   0 Nov04 ?        00:00:03 init
root     29268  29243   0 13:38 pts/0    00:00:00 grep init
```

系統啟動之後，再由 init 行程產生其他行程，因此，可將此行程視為系統的最大行程（PID=1）。

當系統啟動之後，必須隨時監視是否有使用者登入，或其他客戶端的要求服務（如伺服器）。當客戶端（或使用者登入）要求服務時，系統會立即產生另一個『子程序』（Sub-process），來專屬服務該客戶。本書利用一個使用者（tsnien），透過網路登入系統作為範例，來探討系統由啟動後，一直到接受服務的過程如何。圖 12-5 是當系統啟動後立即產生 xientd 超級服務程式，由它來監視是否有使用者登入系統；待使用者登入後，系統會立即產生 in.telnetd 程序來處理網路連線的工作，並產生 login 程序來等待使用者輸入命令，之間過程所產生的程序如下：

```
$ ps –ef                 [執行命令]
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Nov04 ?        00:00:03 init
....
Root        1682       1  0 Nov04 ?        00:00:00
xinetd –stayalive –reuse –pidfil
....
Root       29402    1682  0 19:32 ?        00:00:00 in.telnetd: 220-133-42-19
....
Root       29403  29402  0 19:32 ?        00:00:00 [login]
.....
```

```
tsnien    29404 29403   0 19:32 pts/0    00:00:00 –bash
tsnien    29438 29404   0 19:53 pts/0    00:00:00 ps –ef
```

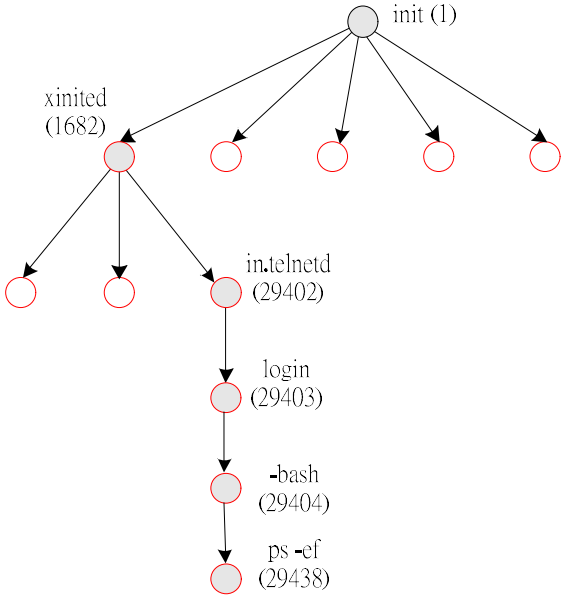

圖 7-3 啟動後登入範例

7-4 指定行程執行

一般使用者可利用 at 命令指定行程的執行時間，也可利用 batch 建立一個批次行程，以下將分別介紹之。

7-4-1 指定行程管理

首先必須先觀察指定行程服務 atd 是否啟動，測試如下：(執行 # systemctl status atd 命令，如沒啟動則鍵入 #system start atd 命令)

```
# systemctl   status   atd     [執行命令]
● atd.service - Job spooling tools
   Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled; vendor preset:
enabled)
   Active: active (running) since 二 2017-02-07 08:48:44 CST; 1min 31s ago
 Main PID: 998 (atd)
   CGroup: /system.slice/atd.service
           └─998 /usr/sbin/atd -f
```

使用者可以由鍵盤上輸入某命令，並指定某一時間來執行該命令，相關命令有：

- at：指定某一時間執行某一命令（程序）。

- batch：批次行程。可指定多個命令同時執行。
- atq：列出使用者所下的 at 命令。
- atrm：刪除 at 所下的命令。

並非任何人都可以指定命令執行，而是必須經由系統管理者授權才行，有兩個檔案可以管理使用者執行 at 命令：

- /etc/at.allow：該檔案內儲存允許執行 at 命令的使用者名稱。
- /etc/at.deny：該檔案內儲存不允許執行 at 命令的使用者名稱。

乍看之下，似乎兩個檔案互有相衝突，其實並非兩個檔案都必須同時存在。若是系統上兩個檔案都不存在的話，表示系統沒有設限，任何人都可以下 at 命令。另外，只有在 at.allow 存在時，編寫該檔案的使用者才可以執行 at 命令，其他使用者則無權執行；反之若存在 at.deny 表示僅該檔案內的使用者不可以執行 at 命令，其他使用者都可以執行。由此可見，allow 機制是採用較嚴謹的管理方式，而 deny 則較寬鬆。至於系統到底要採用哪一種管理方式，這可視環境因素再由管理者（root）自行制定。若是兩個檔案都存在，一般系統可能只會用到較寬鬆的 deny 檔案內容。若當執行 at 命令而所設定的時間還未到，則系統會將所預備執行的命令儲存於：

- /var/spool/at/spool 目錄內。

at 的命令格式如下：

```
$ at [-V] [-q queue] [-f file] [-mldbv] TIME
$ at -c job [job...]
```

常用選項有：

- -V ：印出版本號碼。
- -q queue：使用某一執行佇列（Queue）。

- -f file：讀入執行命令檔案。
- -l ：同 atq 功能。
- -d ：同 atrm 功能。
- -v ：顯示執行時間。
- TIME：指定時間，格式有：
  - HH:MM：指定下達命令當天的時與分，如 17:30 或 1730。
  - HH:MM MM/DD/YY：指定執行命令的年、月、日、時、分，如 17:30 11/12/2004，其中日期也可以 MMDDYY 或 DD.MM.YY 表示。
  - 1am tomorrow：明天早上 1 點。
  - now + count time-units：現在時間以後的時（hours）、分（minutes）、日（days）、星期（weeks）。
- -c ：顯示 at 所下達的命令。

以下用幾種操作模式來介紹 at 命令的使用方法，如下：

7-4-2 指定行程命令 - at

(A) 指定行程操作

直接設定某一特定時間，執行一序列的命令或程式，範例如下：

```
$ date                                  [查閱目前時間，上午 10:38]
一  2 月  6 10:38:42 CST 2017
$ at 10:42                              [指定執行時間，上午 10:42]
at> cp /etc/passwd  /home/student01/at_file      [輸入執行命令]
```

```
at> ^D                                  [結束輸入命令]
[1]+  Stopped                           at 10:42
$ at -l                                 [查閱所建立行程]
1       Mon Feb  6 10:42:00 2017 a student01
$ atq                                   [查閱所建立行程]
1       Mon Feb  6 10:42:00 2017 a student01
$at  –c  1                              [查閱行程內容]
#!/bin/sh
# atrun uid=1000 gid=1000
# mail student01 0
umask 2
XDG_SESSION_ID=1; export XDG_SESSION_ID
HOSTNAME=serCourse; export HOSTNAME
SELINUX_ROLE_REQUESTED=; export SELINUX_ROLE_REQUESTED
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=120.118.165.107\ 49481\ 22; export SSH_CLIENT
....
```

命令 # at 10:40am 表示指定當天早上 10 點 40 分執行命令。下達命令後，首先系統會出現詢問是要在哪一個 Shell 環境執行命令，內定值是原來登入的 Shell 環境（/bin/sh）；接著系統出現 at> 提示符號，使用者可以連續輸入所欲執行的命令或程式；輸入完成，則鍵入『Ctrl +d』（按住 Ctrl 鍵再按入 d 鍵）來停止輸入（將出現 a t>?），並將所設定的命令存入 spool 排序器內。輸入完命令之後，可利用 # at –l 或 # atq 觀察執行後的結果。值得注意的是，at 所指定的命令，一旦時間到達並且已執行完畢，該命令集將由佇列排序器（spool）移除，且不再出現；另外所設定的時間不可以與目前時間太過接近，否則可能會永遠執行不到，這是分時系統所可能出現的問題。

(B) 顯示指定行程

當使用者輸入 at 命令之後，系統會記錄所欲執行批次檔案的時間，並以佇列編號表示；如欲刪除某一 at 批次（atrm 命令）命令的話，也需指定一個佇列編號，操作範例如下：

```
$ at  -l                [顯示指定行程]
81      2005-10-08 15:00 a tsnien
31      2005-05-25 11:32 a tsnien
```

7-4-3 刪除指定行程 - atrm

可利用 atrm 刪除行程，操作範例如下：

```
$ atrm   31        [刪除指定行程]
$ at -l
81          2005-10-08 15:00 a tsnien
```

7-4-4 批次處理 – batch

其實 batch 命令格式與 at 幾乎相同，唯一不同的是 batch 可選擇是否指定時間，而 at 命令一定要指定時間，但兩者所產生的都是一個批次命令(包含一連串序列的命令)。如 batch 沒有指定時間的話，則會自動利用主機較空閒的時機來執行該批次檔案。命令格式如下：

```
$ batch [-V] [-q queue] [-f file] [-mv] [TIME]
```

操作範例如下：(與 at 命令相同，不再另述)

```
$ batch
warning: commands will be executed using (in order) a) $SHELL b) login shell
c)
/bin/sh
at> cp file_21 file_22
at> rm file_21
at> ?
[2]+   Stopped                 batch
$ at   -l
5           2004-11-12 09:23 b tsnien
```

7-5 週期性行程

7-5-1 cron 相關檔案

所謂『週期性行程』(稱之 cron)，就是在設定特定的週期性時間，執行某一批次命令，並以行程稱之。所謂『週期性時間』，表示週期性並重複出現的時間，譬如每天下午 3 點整、每星期六的下午 5 點整、或每月的 15 日上午 10: 30 等等皆是。設定週期性行程對於自動化管理非常有幫助，我們可以請系統在週期性特定時間，去執行某些例行性的任務，或監視

某一共享資源等等。首先須觀察 crond 服務程式是否啟動，如下：(執行 #systemctl status|stop|start crond 命令)

```
# systemctl status crond [執行命令]
● crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor preset: enabled)
   Active: active (running) since 二 2017-02-07 08:48:43 CST; 7min ago
 Main PID: 994 (crond)
   CGroup: /system.slice/crond.service
           └─994 /usr/sbin/crond -n
```

與 cron 相關的有下列檔案，並依照其功能分別說明之，如下：

- /etc/cron.d/：使用者建立的 crontab 檔案儲存目錄。
- /etc/cron.daily/：系統每日執行行程的儲存目錄。
- /etc/cron.hourly/：系統每小時執行的儲存目錄。
- /etc/monthly/：系統每月執行的儲存目錄。
- /etc/weekly/：系統每周執行的儲存目錄
- /etc/cron.allow、/etc/cron.deny：授權使用者操作 cron。
- /etc/cronttab：crontab 示範格式。
- /var/spool/cron/：cron 排序。

如同 at 命令一樣，cron 也是利用 allow 與 deny (/etc/cron.allow 與 /etc/cron.deny) 兩檔案來規劃哪些使用者可以設定 cron。編輯 cron.allow 範例如下：(root 管理者)

```
# cat /etc/cron.allow
tsnien
nien
```

上述範例表示，僅 tsnien 與 nien 使用者可以使用 cron 命令，其他未列入的使用者則

沒有權限設定 cron。

7-5-2 週期執行表格 - crontab

cron 是利用建立一個 crontab 表格來規劃哪一個週期時間，應該執行何種命令，建立的方法可利用 vi 編輯，也可直接利用 crontab 命令操作。以系統所提供 /etc/crontab 為例，來說明其格式：(執行 # cat /etc/crontab 命令)

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# For details see man 4 crontabs
# Example of job definition:
# .---------------- minute (0 - 59)
#|   .------------- hour (0 - 23)
#|   |   .---------- day of month (1 - 31)
#|   |   |   .------- month (1 - 12) OR jan,feb,mar,apr ...
#|   |   |   |   .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
#|   |   |   |   |
#*   *   *   *   * user-name   command to be executed
```

前面四行為設定 cron 的執行環境，其中若每行的最前面是井號 (#) 的話，表示該行為註解說明。接下來，每一行表示一項定期執行的工作，它有 6 個欄位，每個欄位功能如下 (未指定內容以 * 號表示) :

- 第一欄位：表示分鐘，從 00 到 59。
- 第二欄位：表示點 (時) 鐘，從 01 到 24。
- 第三欄位：表示日，從 01 到 31。
- 第四欄位：表示月，從 01 到 12。
- 第五欄位：表示星期，從 0 到 6，其中 0 為星期日。
- 第六欄位：執行命令。

每一行表示一個週期性行程，共有 6 個欄位，之間以空格區分。欄位內特殊符號有：

- *(星號)：代表任何時刻都接受。
- ,(逗號)：表可接受多個時間，譬如分鐘是 15, 30, 45，則 15 分、30 分與 45 分都接受。
- -(減號)：表示時間區段，譬如日期是 2-10，則表示 2 到 10 之間每天都接受。
- /n(斜線)：表示每隔 n，譬如分鐘是 */5，則表示每隔 5 分鐘。

7-5-3 建立週期命令 - crontab

建立與操作 cron 表格，係利用 crontab 命令來達成，格式如下：

```
$ crontab [ -u user ] file
$ crontab [ -u user ] { -l | -r | -e }
```

常用選項如下：

- -u user：指定某一使用者的 crontab 表格，一般都是由系統管理者設定。
- file：直接導入 crontab 檔案。
- -l ：列出使用者在 crontab 所設定的定期命令。
- -r ：刪除 crontab 命令。
- -e ：直接進入 vi 編輯 /etc/crontab 檔案。

執行 **$ crontab –e** 之後，會立即進入 vi 編輯模式，並自動編輯使用者的 cron 表格，輸入內容後直接儲存並離開，且不要指定檔案名稱。操作範例如下：(以 tsnien 名稱登入並執行 **$ crontab -e**，進入 vi 模式下編輯)

```
# Test cron of tsnien
1 23 1,15 * * fsck /home
1,30 * * * * quota –a
```

7-5-4 建立 crontab 範例

假設我們希望系統能週期性的執行下列工作：

✧ **每個月的 1 與 15 日**，針對 /home 做檔案系統檢查一次。

✧ **每 30 分鐘檢**查一次所有使用者的配額是否有溢滿。

(A) 建立 crontab

執行 **$ crontab –e** 之後，會立即進入 vi 編輯模式，並自動編輯使用者的 cron 表格，輸入內容後直接儲存並離開，且不要指定檔案名稱。操作範例如下：(以 tsnien 名稱登入並執行 **$ crontab -e**，進入 vi 模式下編輯)

```
# Test cron of tsnien
1 23 1,15 * * fsck /home
1,30 * * * * quota –a
```

第一行為每月 1 與 15 日的 23:01 時執行 fsck /home 命令；第二行為每小時的 1 與 30 分時執行 quota -a。值得注意的是，兩個都是系統管理命令，所以 tsnien 必須是 root 群組的成員才可以操作。儲存並離開後，可觀察是否建立成功，操作範例如下 ($ crontab -l)：

```
[tsnien@linux-1 tsnien]$ crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.4838 installed on Mon Oct 10 10:09:03 2005)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
# Test cron of tsnien
1 23 1,15 * * fsck /home
1,30 * * * * quota –l
```

(B) crontab 表格儲存

每一個使用者僅能建立一個 cron 表格，第一次建立時系統會將其儲存於 /var/spool/cron/ 目錄下，並以使用者名稱 (如 tsnien) 儲存。如果使用者欲修改其內容，則需再執行 crontab –e 命令，系統就會將原使用者的表格再叫出來編輯。接下來，觀察剛才

tsnien 所建立的表格：(需具 root 權限)

```
# cat   /var/spool/cron/tsnien
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# Test cron of tsnien
1 23 1,15 * * fsck /home
1,30 * * * * quota -l
```

7-5-5 檔案導入與刪除

(A) 檔案導入

我們也可以先利用 vi (或其他編輯工具) 建立一個 cron 檔案，再將其導入成 cron 表格；假設期望建立的週期性工作如下：

- ✧ 每天 09:30 時執行 who > who.day 命令一次。
- ✧ 每星期一的 17:30 時執行 ps –ef > ps.week 命令一次。

則可利用 vi 編輯$ vi cron_1，檔案內容如下：

```
$ cat   cron_1          [觀察 cron_1 檔案內容]
# file input cron table
# file name: cron_1
30 09 * * * who >who.day
30 17 * * 1 ps -ef >ps.week
```

接下來，再將 cron_1 導入 cron 表格內，操作範例如下 ($crontab cron_1)：

```
$ crontab   cron_1                  [載入 cron_1 檔]
$ crontab   -l                      [觀察 crontab 內容]
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# file input cron table
# file name: cron_1
30 09 * * * who >who.day
30 17 * * 1 ps -ef >ps.week
```

不過有一點必須特別注意，假使先前已經建立過 cron 表格，而且仍存在系統，此時再導入其他 cron 檔案時，就會將原來的 cron 表格覆蓋，並以新表格取代之。

(B) 刪除 crontab

使用者也可以刪除自己的 cron 表格 (crontab -r)，操作範例如下：

```
$ crontab   -r              [刪除]
$ crontab   -l              [觀察 crontab 內容]
no crontab for tsnien
```

7-6 系統服務

7-6-1 何謂服務？

所謂『服務』？就是某一行程所能完成的特殊功能，而此功能具有共通性可以被其他行程或使用者所引用；當使用者 (或其他行程) 需要類似的功能時，可呼叫或請求某一行程來完成此功能，一般將此行程稱之為『服務』。基本上，系統內必須存在許多不同的行程，分別提供不同的服務需求。再說建構一套作業系統的目的，是要提供各種介面，讓使用者方便操作電腦；為了達成此目的，作業系統必須提供各式各樣的『服務』(Service) 讓使用者使用。簡單的說，『服務』即是提供某種功能的程式，譬如印表列印、磁碟機存取、網頁存取、郵件轉送、甚至關機服務等等。

執行中的系統到底需要哪些服務？這是值得推敲的問題。其實需要哪些服務並沒有一定的規範，而是要看該系統所欲扮演的角色有無絕對性的關係。譬如說，僅要當作一部單人使用工作站，與要當作網路伺服器的系統，他們所欲啟動的服務程式之差異性就非常大。以下將介紹系統所欲執行的模式，應該啟動哪些服務？與應如何啟動。

系統到底安裝了些服務，在何種情況下要啟動哪些服務，早期 Unix system V(SysV) 版本採用 init 啟動方式，它的做法是將所有已安裝的服務程式儲存於 /etc/rc.d/init.d/ 目錄下，系統啟動模式有 runlevel_0 ~ runlevel_6 等七個等級，可依照使用者選擇啟動哪一個等級。

啟動哪一個等級，會執行哪一目錄下服務，則在 /etc/inittab 檔案規劃，並將各等級的服務程式儲存於 /etc/rc0.d/～/etc/rc6.d/ 等目錄下，當然這些檔案皆以鏈路(link) 方式，索引到 /etc/rc.d/init.d/ 目錄下檔案。系統啟動時，依照所選定的 runlevel 等級，執行該相對目錄下服務程式後，再執行 /etc/rc.d/local 檔案，該檔案提供除了系統服務之外，其它新增的服務或系統操作命令，譬如增加路由表、防火牆規則等等。

其實，服務程式也是屬於主從式架構(Client/Server Architecture)，是被動的，它不停的監督掃描是否有客戶端提出要求，如果有則立即給予服務，因此 SysV 稱此監督程式為 Daemon。SysV 提供『獨立模式』(Stand alone Daemon)與『超級監督』(Super Daemon)等兩種模式，前者是由服務程式自行監督是否有服務需求，後者是利用一只專屬監督程式(xinetd 或 inetd)負責。xinetd 負責多個服務程式，掃描相對應服務埠口(socket Port)發現有服務需求時，再呼叫啟動該相對的服務。

自從 CentOS 7 (RedHat 6 新版本)之後便拋棄了原 SysV 的啟動模式，雖然已沒有 runlevel 等級選擇，但她還是保留有向下相容能力。各 runlevel 執行的程式也將其歸納整合，可選擇不同的 runlevel，其實是執行相同的服務程式。新版本為了應付新的多核心主機架構，認為執行服務程式應該可以拋棄原上下從屬的關係，各服務程式應該可以獨立執行。當然許多情況下，服務程式是有從屬關係，但如發現執行某一程式，必須先執行另一程式時，則再去啟動它就好。如此，就可以讓多個核心同時執行不同的服務程式，如此就可以增加系統的執行速度。因此，新版本不再有 xinetd 超級監督程式，每一個服務程式皆是 Stand alone。

7-6-2 服務相關檔案

系統服務相關檔案：

- **/usr/lib/systemd/system/**：系統已安裝的服務程式，有點類似 SysV 統的 /etc/rc.d/init.d/ 目錄下得的服務程式。

```
# ls   /usr/lib/systemd/system
abrt-ccpp.service                       numad.service
abrtd.service                           oddjobd.service
abrt-oops.service                       packagekit-offline-update.service
abrt-pstoreoops.service                 packagekit.service
abrt-vmcore.service                     paths.target
abrt-xorg.service                       plymouth-halt.service
accounts-daemon.service                 plymouth-kexec.service
alsa-restore.service                    plymouth-poweroff.service
alsa-state.service                      plymouth-quit.service
alsa-store.service                      plymouth-quit-wait.service
anaconda-direct.service                 plymouth-read-write.service
```

✧ **/run/systemd/system/**：系統執行中產生的服務程式，會隨著原程式結束而結束。

```
# ls /run/systemd/system
session-1.scope   session-1.scope.d   session-c1.scope   session-c1.scope.d
```

✧ **/etc/sysconfig/**：各服務的監督檔案(許多已不再使用)。

```
# ls /etc/sysconfig
atd                 firewalld             libvirt-guests     qemu-ga
samba
authconfig          grub                  man-db             radvd
saslauthd
autofs              init                  modules            raid-check
selinux
cbq                 ip6tables-config      netconsole         rdisc
smartmontools
cgred               iptables-config       network            readonly-root   sshd
console             irqbalance            network-scripts    rpcbind         sysstat
....
```

✧ **/etc/systemd/system/**：系統啟動時，執行的服務程式，有點像以前 runlevel 的 /etc/rc0.d/ ~ /etc/rc6.d/ 底下的服務程式，但現在已沒有分等級了。(可利用 setup 命令選擇)

```
# ls   /etc/systemd/system
basic.target.wants                          getty.target.wants
bluetooth.target.wants                      graphical.target.wants
dbus-org.bluez.service                      multi-user.target.wants
dbus-org.freedesktop.Avahi.service          printer.target.wants
dbus-org.freedesktop.ModemManager1.service  remote-fs.target.wants
....
```

7-6-3 服務的類型

早期 SysV 將比較小的服務程式歸納由 Super Daemon 來監視，新版本拋棄它之後，即將服務區分多個類型，方便管理與監督執行。由 /usr/lib/system/system/ 目錄下可以觀察到服務程式的命名是『名稱.類型』(name.unit)，譬如：

```
# ls /usr/lib/systemd/system   | grep   ^atd*
atd.service                          [service 類型]
```

各服務類型說明如下：

- [.service]：服務類型(service unit)。最普遍的服務程式，一般伺服器皆屬於此類型。
- [.socket]：插槽類型(socket unit)。透過 socket 掃描通訊連結的服務。
- [.target]：目標類型(target unit)。它屬於依照某一目的，整合多個服務程式成為一只服務的類型，
- [.mount/automount]：掛載類型(mount unit)：以掛載檔案系統相關的服務。
- [.path]：路徑類型(path unit)：與搜尋與偵測檔案系統路徑有關的服務。
- [.time]：時間類型(time unit)：與時間或週期性有關的服務。

各類型的監督掃描方式有所不同，它們之間的運作方式，需再參閱其他書籍瞭解。

7-7 服務管理 – systemctl

除了將服務命令嵌入 Runlevel 的啟動目錄之外，也許還需要隨時啟動、停止或檢查某一個服務，因此需要一些管理命令來輔助。直接執行服務程式是最簡單的管理工具，其命令格式如下：

```
# systemctl   {start|stop|restart|reload|condrestart|status}   [service unit]
```

其中：

- service unit：為 /usr/lib/system/system/ 目錄下的服務程式，譬如：sshd、apmd、nfs、squi、httpd、firewalld 等等。

■ 命令彙集如下： (以 httpd 服務為例)

功 能	命令格式
查詢目前運行中服務	# systemctl
顯示服務狀態	# systemctl status httpd
啟動服務	# systemctl start httpd
停止服務	# systemctl stop httpd
重新啟動服務	# systemctl restart httpd
重新導入服務	# systemctl reload httpd
開機時自動被啟動	# systemctl enable httpd
開機時不被啟動	# systemctl disable httpd
目前是否執行中	# systemctl is-active httpd
開機時是否被啟動	# systemctl is-enable httpd

■ 範例：

```
# systemctl  is-active  atd        [查詢 atd 是否啟動]
active
# systemctl  is-enabled  atd       [查詢 atd 下次開機是否被啟動]
enabled
# systemctl  stop  atd             [停止 atd 執行]
# systemctl  is-active  atd        [查詢 atd 是否啟動]
inactive
# systemctl  start  atd            [啟動 atd 服務]
# systemctl  is-active  atd        [查詢 atd 是否啟動]
```

```
active
# systemctl   restart   atd        [重新啟動 atd 服務]
# systemctl   is-active   atd      [查詢 atd 是否啟動]
active
```

習　題

1. 何謂『行程』(Process) ？
2. 請敘述『行程』與『程式』之間的關係如何？(一對多、多對一或多對多)
3. 何謂行程『孤兒』(Orphan) ？
4. 請比較『前景執行』與『背景執行』之間有何不同的地方？
5. 請敘述『監督行程』(Daemon) 的運作情形如何？
6. 請說明 at.allow 與 at.deny 兩檔案的功能為何？如果兩檔案都存在的話，系統又如何判斷？
7. 請敘述 # at 09:30 –f file_1 命令之功能為何？
8. 請敘述 # batch –f file_1 命令的功能為何？
9. 請敘述 # crontab file_1 命令之功能為何？
10. 請舉例說明，如何刪除一個已下達的 at 行程？
11. 何謂『週期性行程』(cron) ？
12. 請說明 cron.allow 與 cron.deny 兩檔案的功能為何？如果兩檔案都存在的話，系統又如何判斷？
13. 在 Unix/Linux 系統下，行程的優先權與執行時間的關聯如何？提高行程優先權，是否完成時間就會較快？請敘述其原因如何？
14. 何謂『服務』(Service) ？
15. 請分別說明『常駐性』與『暫時性』服務程式的特性如何？
16. 請分別說明『阻斷性』與『無阻斷性』服務的特性如何？