



## 1. Cfengine 引言

隨著科技日漸精益求精，對改變引入的成本將會下降。

—Alvin Toffler, *Future Shock*, 1970

Cfengine 是一款可自動對聯網的電腦進行配置和維護的免費套裝軟體。它適用於所有基於UNIX或類UNIX的作業系統，並且它可以通過UNIX相容的環境/庫 Cygwin 在較新版本的Windows作業系統中運行。

Cfengine 適用於管理各種環境，從一台主機到上萬台主機的機群均可使用。到撰寫本書為止，我們現在所知的用於一般性管理的最大安裝機群約為 20,000 台。

Cfengine 可從很多方面對系統配置和維護進行管理，包括以下幾點：

- ◆ 完成後期安裝任務，例如配置網路介面資訊。
- ◆ 編輯系統配置檔以及其他檔。
- ◆ 建立信號連接。
- ◆ 檢驗、更正文件許可及所有權。
- ◆ 刪除無用檔。
- ◆ 壓縮被選檔。
- ◆ 在網路中分發檔。
- ◆ 自動掛載NFS檔系統。
- ◆ 檢查重要檔和檔系統是否存在及其完整性。
- ◆ 執行命令及腳本。
- ◆ 應用安全相關的補丁以及相似系統的修正。
- ◆ 管理系統伺服器進程。

Cfengine 的目的在於執行基於策略的配置管理。從實際的應用角度來講，這意味著 Cfengine 可以最大限度的簡化系統配置及維護任務。例如：要優化一個特定系統，用戶不再需要使用 Perl 或其他用戶習慣的 shell 來編譯一個程式來執行每項要求，取而代之的是，用戶可以通過寫一個更加簡單的策略來描述用戶希望自己的主機如何被配置。Cfengine 軟體可根據這些描述來決定哪些執行方式和/或補救方法是需要被完成的。這些策略描述也可以用於確保系統的配置能如系統管理員所希望的一樣被保持下來。

以下是關於這樣一個策略描述帶簡要注解的例子：

## 策略範例 1：Cfengine 配置介紹

```
control:                                     一般指示
    tmpdirs = ( tmp:scratch:scratch2 )      定義變數列表
    actionsequence = ( files copy tidy )    指定執行動作 (按順序的)

files:                                       檔所有權以及保護規範
    /usr/local/bin owner=root group=bin mode=755 action=fixall recurse=1

copy:                                       複製檔到本地系統
    solaris::                               僅應用於Solaris系統
        /config/pam/solaris server=pammaster dest=/etc/pam.d recurse=1
    linux::                                  僅應用於Linux系統
        /config/pam/common-auth server=pammaster
        dest=/etc/pam.d/common-auth

tidy:                                       管理臨時草稿目錄
    /${tmpdirs} include=* age=7 recurse=inf
```

這個簡單的配置檔被分為四個小節，每一小節又以一個冒號為結尾的關鍵字引入，分別是**控制**：**(control:)**，**文件**：**(files:)**，**複製**：**(copy:)**，以及**清理**：**(tidy:)**。

**控制 (control)** 小節定義了一系列目錄，我們將其命名為tmpdirs，這個我們將在以後中用到（在**清理 (tidy:)** 小節）。

**檔 (files)** 小節指定了所有在/usr/local/bin 目錄裏的檔都應被根用戶 root 和 bin 組所有，並且定義了檔的模式為 0755（所有權者擁有所有權限，其他用戶擁有讀操作及執行操作）。當 Cfengine 運行時，它將修改所有與該配置檔描述不相符的屬性以及許可權。因此，該小節用於定義在本地二進位目錄中適合可執行檔的所有權以及許可。

**複製 (copy)** 小節中描述了對 Linux 和 Solaris 系統的不同配置方法。在 Solaris 系統中，當位於主伺服器上/config/pam/solaris 目錄中的檔更新後，位於/etc/pam.d 目錄中的檔也會隨之更新。而在 Linux 系統中，只有檔/etc/pam.d/common-auth 會隨著 PAM 的主配置檔的更新而更新<sup>1</sup>。需要注意的是：這兩個規範執行著相同的底層系統配置以及維護策略：當需要的時候，相關的 PAM 配置檔將從主伺服器中被更新。

1. 這樣做的原因並不是非常顯而易見。很多 Linux 系統使用包含檔機制的 PAM 來傳播本檔的 PAM 棧數到其他所有的可用 PAM 服務的配置檔。因此，只有這個主文件會不斷改變。

最後，**清潔 (tidy)** 小節舉例說明了隱式迴圈的利用。例子中單一的指示應用於 `tmpdirs` 列表中的每一個目錄檔。對於每一個目錄，`Cfengine` 會刪除在目錄中所有的項，或者子目錄中任意一個七天內沒有被使用過的專案（包括那些以點號開始的檔案名）。如同在這個示例配置檔中的其他方針，這一節執行的策略是：刪除一周內未被使用的臨時目錄中的專案。

所有的 `Cfengine` 配置檔在以上所介紹的方面以及其他相似元素方面的描述都有所不同，有些元素的描述更加細緻具體。在進一步介紹關於使用 `cfengine` 的更多技術細節之前，接下來將要介紹的是一些最重要的基礎知識以及理論指導思想。

## 1.1 基本原理概念

如前所述，`cfengine` 在主機上運行以使他們的配置符合特定的策略。以下是這些術語的正式定義：

**定義 1：主機 (host)**。一般來講主機是一台運行於作業系統（如 UNIX, Linux, Windows）之上的一台獨立的電腦。有時我們也認為其是一台機器，主機也可以是如 VMware 或 Xen/Linux 環境所支援的虛擬機。

**定義 2：策略 (policy)**。策略是我們希望主機以怎樣的方式運行的一種規定。與其他任何電腦程式不同的是，一個策略實質上是用於描述技術細節和特徵的基本文檔。`Cfengine` 實施經過我們考慮的策略。

**定義 3：配置 (Configuration)**。一台主機的配置檔是它資源的實際狀態，例如，檔的許可權和內容，已安裝軟體的詳細目錄等等。配置檔也指某一特定的主機在指定時間的事務狀態。

我們使用 `cfengine` 的目標是什麼呢？答案是，策略一致性配置。我們想要對一台或更多的主機制定一套規範，來描述他們的特性以及他們間的相互作用（或許是用來解決一個商業問題），然後我們就要把細節，執行和維護交給自動控制代理去處理：`cfagent`。

人類善於理解輸入和思考解決辦法，但是在執行方面卻不是非常靠得住 (**doing**)。機器和軟體代理善於並可靠地執行任務，但不善於理解或找到實際的解決問題的辦法。通過使用 `cfengine`，你可以令“人-機器”這樣組織的不同部分專注於它們各自所擅長的工作。

`Cfengine` 在一個相對較低的水準上工作，因此這是實際上的實現方法而不是概念上的方式。不僅如此，在決定策略之時，你會發現有足夠的高級概念以供參考。

### 1.1.1 承諾，行為和操作

一個cfengine策略可以被認為是系統對某個審計員允諾對其配置的表單。大部分承諾包含了改變主機並使其滿足策略承諾的可能性。我們稱這樣的改變為行為或者操作。和你可能猜想到的一樣，這裏的審計員是cfengine它自身的一部分。如果沒有滿足承諾，Cfagent也是執行改變系統的技工或者外科醫生。

通過以這樣的方式描述它的操作，配置管理可以被認為是一種服務，該種服務與監控和維護緊密相關，並且此服務在無需將一個系統與核心許可權相關聯的情況下就可以依據需求而被“購買”到。

**定義4：操作 (Operation)。**一組改變就是一個操作。Cfengine處理對系統的改變，而操作則被嵌入到一個cfengine策略的基本文句當中。這些告訴我們策略是如何約束主機的，換句話說，就是我們怎樣避免主機失控。

以下就是一個關於檔屬性的承諾的例子：

files:

```
/etc/passwd mode=a+r,go-w owner=root group=root action=fixall
```

在該聲明中有隱含的操作(動作)：在特定情況下，如果/當他們不符合這個規定時，這些操作就會改變屬性。

### 1.1.2 收斂性

Cfengine 的一個重要特性就是收斂。這是區分它與一般的電腦語言的重要特性。該特性有助於避免系統產生分歧：即使運行在無法控制的環境下。

**定義5：收斂 (Convergence)。**如果一個操作總是使一台主機的配置接近於理想的、策略一致性的狀態，並且該主機已經處於此種狀態，那麼此操作不會對主機產生任何影響，那麼這個操作就是收斂的。我們可以通過以下的規則從功能型術語方面總結這一點：

Cfengine(incorrect state)  $\rightarrow$  correct state

Cfengine(correct state)  $\rightarrow$  correct state

有時我們會把“正確狀態”稱之為“健康狀態”，以此來隱喻一個配置很差的主機猶如某種疾病所帶來的痛苦。

以下是用於一個 ASCII 檔編輯中的例子：

```
editfiles:  
...  
AppendIfNoSuchLine "Important configuration line"
```

該操作說明當一個文本不在某個指定檔中時，`cfengine`會將該給定的文本附加在該檔的末端。因此，這個策略一致性的配置是在當前文本行已經存在，或者一旦完成添加文本行到指定檔中的操作，任何過多的操作都不會發生。那麼，我們認為操作**AppendIfNoSuchLine**是收斂的。

不要低估了收斂的價值。它提供了穩定性。正是由於 `cfengine` 的語言介面強烈阻止你做出任何“非收斂的”事情，它同樣也有助於避免錯誤的發生。而利用該特性的代價是你必須學著以一種收斂的方式思考，對於大多數剛接觸 `cfengine` 的人來講這是種全新的思想。

### 1.1.3 類與聲明：從一台主機到多台主機

使 `cfengine` 的策略可讀的特性之一是，`cfengine` 有隱藏的由代理機決定所有的複雜需要的決策的能力。為達到這樣的目的，`cfengine` 使用一種聲明語言表達策略。

這種可以被聲明的語言僅僅是一個結構化的句子列表(在`cfengine`中，它是一個策略承諾的列表)。沒有任何特殊的表達次序，它只描述了一個最終的目的。達到該目的的具體方法是不明確的，是由解釋規範的引擎進行評估和實現的。這與`shell`或`Perl`等細緻地控制整個過程的每個步驟之類的程式性或命令性語言形成了對比。

命令性語言專注於過程。而可聲明性語言則關注目的和預測的結果。

在 `cfengine` 中類的使用就是一個例證。類就是一種不需要很多“if-then-else”語句的決策方式。一個類就是一個標誌符，當一個測試結果為真時，該布林型變數的值就是 `true`。換句話說，它隱藏了一個 `if` 測試的結果。以適當的系統並且/或者在恰當的情況之下，一個類是用來限制 `cfengine` 行為的範圍內。

類的好處是所有的測試都可以被隱藏於 `cfengine` 的之內，只有當需要或者如果需要時，結果才是可見的。

**定義 6：類 (classes)**。一個類是將一個或多個主機的複雜環境裁剪和限制在一個區域之內的一種方式，並可以通過一個標誌或名字而被訪問。類對範圍有如下描述：一些事情被限制於其中的地方。

例如，當且僅當 `cfagent` 運行於以 Debian GNU/Linux 為作業系統的主機上時，類 `debian` 的值是真。

#### 1.1.4 自願合作

每個主機保持其個體自治是 `cfengine` 元件的基本特性。管理員可以隨時將主機從基於 `cfengine` 的管理中退出。該原則引出了基本的設計和執行的決定：

**定義7：自治 (Autonomy)**。如果資訊沒有明確地被其自身請求，沒有 `cfengine` 元件能夠接收資訊。

理解這一點很重要。這並不代表主機的集中控制不能實現。集中式控制是大多數管理員選擇使用 `cfengine` 的方式。實際上，若想要實現集中控制，你所需要做的僅僅是為你所有的主機做出策略決定，以便於從核心許可權獲得策略規範。

自治並不意味著如果環境存在一些小組或有特殊需要的次文化的特殊需要，有可能是他們保持其特殊性。任何自我委派的許可權都不能超越他們的本地決定。

那麼，策略來自哪里？任何主機都依據 `cfengine` 可以找到本地目錄（通常是類似於 UNIX 主機之上的 `/var/cfengine/inputs`）的一個策略規格來工作。如果希望主機從某個核心管理器或授權獲取控制，那策略必須包含啟動規格說明，即“我決定，我要下載且遵守中心管理器的策略規範。”

任何時刻，每個主機都可以取消該策略決定。這是 `cfengine` 安全模式的核心部分。

#### 1.1.5 可測量性

`Cfengine` 是按照最大的可測量性而設計的。它的可測量性至少和其他任何系統一樣優秀，因為它允許最大限度的工作量的分配。

**定義8：可測量的工作分配 (Scalable distributed action)**。每一個主機可基於自己的本地策略副本，對檢查和維護自身的操作負責。

這不意味著你錯誤的決策也行得通。舉例來說，當你要求一萬台主機同時從同一處獲取某些資訊的時候，網路服務始終會成為其瓶頸。

實際上每一個 `Cfengine` 代理保持一份本地策略的副本（不論它是在本地編寫或是繼承

於一個核心授權)，這意味著即使網路通信出現故障，Cfengine 也將持續工作。

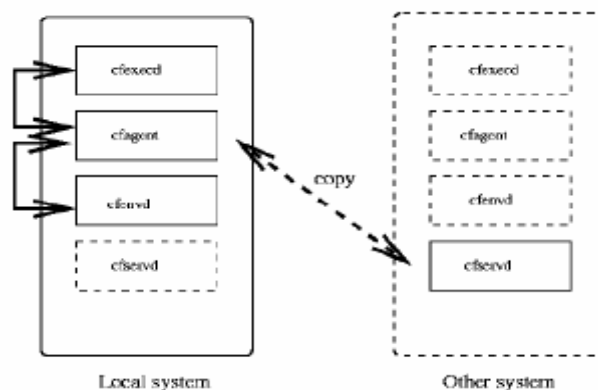
## 1.2 Cfengine 的組成部分

Cfengine 軟體由很多部分組成，它們是一些共同運行的獨立程式。(見圖表 1.1)<sup>2</sup>

Cfengine 的這些組成部分是：

- ◆ Cfagent: 解釋策略的承諾並且以收斂的方式執行它們。代理可使用由統計監測引擎 Cfenvd 產生的資料，並且它能從運行于本地或遠端主機上的 Cfenvd 中獲取資料。
- ◆ Cfexecd: 執行 Cfagent，並且記錄它的輸出(可選擇通過電子郵件寄出摘要)。它可以在一個後臺程式(standalone)的模式下運行，或者可以通過 cron 在一個類似於 Unix 的系統上運行。
- ◆ Cfservd: 監控 Cfengine 的埠：提供檔資料，並在接收一個來自 cfrun 的連接的基礎上啓動 Cfagent。請注意，沒有資料可以通過這個後臺程式。
- ◆ Cfrun: 聯接遠端主機，並要求他們運行 cfagent。
- ◆ Cfenvd: 收集在每台主機上使用資源的統計資料，用於異常狀況的檢測。資訊以 cfengine 類的形式被代理獲得，因此代理可以及時地對異常的動態狀況進行檢查並作出反應。
- ◆ Cfkey: 在主機生成“公有-私有”密鑰對。一般作為 Cfengine 軟體安裝過程中的一個步驟，你只需運行一次該程式。
- ◆ Cfshow: 一旦你對它的內部存儲感興趣，cfshow 便將 cfagent 的資料庫內容以 ASCII 的格式導入。
- ◆ Cfenvgraph: 將 Cfenvd 的資料庫內容導入為一種可用於圖示格式顯示一台主機在其環境中的一般行爲。

2. 第一版與第二版之間的組成部分是不同的。因為 cfengine 版本 1 已經不再被支持，在此我們只討論 cfengine 2，並且我們強烈建議你使用版本 2。此外，在撰寫本書的同時，cfengine 的第 3 版正在完善中，但是想要它完全取代第二版還是需要很多年的時間。第三版會吸取所有基於第一與第二版的經驗教訓，並結合網路與系統管理研究的發展動態，逐步完善起來。



圖表 1.1 Cfengine 的組成部分以及它們之間的聯繫

圖表 1.1 舉例說明了在不同主機上 cfengine 組成部分之間的關係。在一個給定的系統裏，cfexecd 程式將開始運行 cfagent；它還會在 cfagent 運行中處理日誌記錄。此外，cfagent 會在本地系統啓動運行如主機間的檔複製的操作，並且他們以 cfserverd 系統來取得在遠端系統上的資料。

## 1.3 開始安裝

在這一節中，我們將要安裝並運行 cfengine。在理解編譯語言的細節之前，首先應該讓 cfengine 的組成運行於一個基本、簡單的策略之下，使得引擎慢慢運作起來。當你瞭解了它的操作方法之後，便可以一步一步建立起自己的策略。

### 1.3.1 建立軟體

Cfengine 的安裝如同其他大部分 UNIX 開源軟體一樣。你可以選擇套裝軟體安裝版本配置到你的系統，或者從源代碼自己編譯。

在任何一種情況下，你都需要兩個庫：BerkeleyDB，用於內部資料庫的使用；和 OpenSSL，用於加密方法。這些庫都是開放資源，並且如同 cfengine 一樣可以免費使用。沒有這些庫，你將無法使用 cfengine，你也不能用其他的庫來替代這兩個庫。<sup>3</sup>

3 · cfengine 所使用的資料庫是基於記憶體的高速低級別的運行結構。他們不能用於用戶資料存儲。確切的說，cfengine 需要一個有著很強查找能力的本地資料庫。因此，SQL 相關的資料庫是不適用並且也不可能用於 cfengine。

開始安裝cfengine的第一步，需要從<http://www.cfengine.org>，或者其鏡像站點的任意一處下載源代碼文件。下載的包將是一個以cfengine-2.x.x.tar.gz命名的壓縮包，其中x.x是指cfengine第二版的尾碼版本號。

以下的幾步總結了建立cfengine所需要的大體步驟：

### 步驟1：從開源代碼安裝Cfengine

```
$ tar xzf cfengine-2.x.x.tar.gz
$ cd ./cfengine-2.x.x
$ ./configure
$ make
$ sudo make install
```

在類UNIX用戶上安裝二進位碼的默認位置是在/usr/local/sbin下。

這個目錄有時是一個共用檔系統（例如通過NFS而安置的遠端檔系統）。在網路失去連接而Cfengine一定要可以繼續運行時，這將成爲一個問題。爲此，cfengine的目錄樹可以繼續維持cfengine的二進位碼的複製。

## 1.3.2 建立你的第一個cfengine主機

終於，你可以讓cfengine完成在一個新的電腦系統上安裝本身的大部分工作。作爲一個初學者，你應該希望學習這一項任務是如何實現的。爲了實現這個目標，我們將手動安裝cfengine，這樣每一步驟將顯而易見。

爲了避免不需要的網路檔系統間的關聯性，cfengine使用了同一個目錄以確保其可以安裝于任一主機（除了無磁片的用戶端）。其默認安裝目錄（通常稱爲“工作目錄”）是/var/cfengine。在這裏，我們可以假定cfengine的可執行檔被安裝在/usr/local/sbin。

接下來的一步是建立cfengine工作目錄樹的基本結構：

### 步驟2：手動建立Cfengine的工作目錄

```
# mkdir /var/cfengine
# mkdir /var/cfengine/bin
# mkdir /var/cfengine/inputs
```

接下來，在工作目錄`bin`的子目錄下（例如：`/var/cfengine/bin`）建立`cfengine`可執行程式的本地副本。實際運行過程中，是這些副本被執行，因此當網路在執行任務期間斷掉，也不會對系統產生風險。

### 步驟3 複製Cfengine二進位碼到工作目錄

```
# cp /usr/local/sbin/cfagent /var/cfengine/bin
# cp /usr/local/sbin/cfexecd /var/cfengine/bin
# cp /usr/local/sbin/cfserverd /var/cfengine/bin
# chown -R root:0 /var/cfengine
# chmod -R 755 /var/cfengine
```

現在，這些二進位碼被放在了一個可靠的位置中，接下來我們可以通過建立一個簡單的`cfengine`策略來對這個代理進行測試。

建立如下的文件：`/var/cfengine/inputs/cfagent.conf`。

### 策略範例2：初次測試的簡單策略

```
#/var/cfengine/inputs/cfagent.conf
control:
    actionsequence = ( shellcommands )
shellcommands:
    “/bin/echo Danger, Will Robinson!”
```

這就是你所需要的測試`cfengine`的所有腳本。這個策略非常簡單，其目的在於列印顯示一條資訊。現在可以通過運行這個代理來進行測試。在默認狀態下這個代理會在工作目錄下尋找`cfagent.conf`這個檔。需要注意的是我們必須在第一次運行`cfagent`以前運行一次`cfkey`命令。<sup>4</sup>

### 步驟4：運行代理以測試Cfengine的基本功能

```
# /usr/local/sbin/cfkey 在第一次運行cfagent命令前運行一次該命令
# /var/cfengine/bin/cfagent
cfengine::/bin/echo Dange: Danger, Will Robinson!
```

4 · 這個命令將為本地系統建立公有-私有密鑰對，並在`/var/cfengine/ppkeys`子目錄下存儲結果檔。它也可以在`cfengine`工作目錄下構建`randseed`檔和其他多份附加子目錄。

現在，令人驚奇的事情發生了！在一秒的時間上立即運行**cfagent**命令兩次，你會發現沒有任何事情發生。這是正常的現象。實際上，到離你上次運行**cfengine**的一分鐘時間內，不會有任何其他的事情發生。如果你運行**cfagent -v**，調用詳細模式，你將會看到輸出資訊中包含以下內容：

```
cfengine:: Nothing scheduled for
[shellcommand./bin/echo Danger, W] (0/1 minutes elapsed)
```

這個資訊是告訴你，**cfengin**覺得這樣重複這個承諾動作實在太快了。我們將在後面討論**cfengine**的事務處理鎖時會返回到這個問題上。

現在祝賀你！你已經成功地使用**cfengine**了！

### 1.3.3 建立一個長期的配置

讓**cfengine**運行於一個規則的基礎上是件很正常的事情：每小時運行一次或者每15分鐘運行一次，這完全取決於你的需求。經常性的運行**cfagent**不會對系統造成負擔，因為引擎不會重複多餘的動作（你應該還記得收斂性的特性）。我們可以通過手動編譯一個**crontab**檔來完成這個目標，但是，取而代之的是，現在我們可以讓**cfengine**來為我們完成這個目的。

編輯策略檔使之與下面的例子相匹配：

#### 策略範例3：Cron每15分鐘運行cfexecd

```
# /var/cfengine/inputs/cfagent.conf
control:
    actionsequence = ( editfiles )
    EmailTo = ( sysadmin@mydomain.tld )

editfiles:
    !SuSE::
    { /var/spool/cron/crontabs/root
        AutoCreate
        AppendIfNoSuchLine
        "0,15,30,45 * * * */var/cfengine/bin/cfexecd -F"
    }

    SuSE::
    { /var/spool/cron/tabs/root
```

```
AutoCreate
AppendIfNoSuchLine
“0,15,30,45 * * * * /var/cfengine/bin/cfexecd -F”
}
```

我們已經刪掉了Lost in Space的策略，取代它的是根用戶root的**crontab**檔。現在，這個策略可以執行一些簡單檔的編輯，而不是運行一個shell命令。另外，它對以雙冒號為結尾的SuSE Linux提供了一些參考提示。這是另一種cfengine類的例子，並且他們定義了後續承諾何時會被應用。

在這個例子中，我們考慮到SuSE Linux為**crontab**使用了一個不同於其他大部分作業系統的慣用性目錄的事實。因此在我們的策略檔舉例的後半部分，我們為SuSE用戶指定了一條規則，而對其他非SuSE用戶指定了另一條規則(正如你所猜想的，“!”意味著邏輯非)。

在這兩種情況下，如果在**crontab**中不存在某句話，承諾將會把這句話加入其中。<sup>5</sup> 我們將會在本冊的後續章節中會對檔的編輯做出更多細緻的解釋。

修正後的策略文件也在控制小節中包含了附加的指示說明。在cfexecd開始啟動的時候，它可以將由cfagent所產生的目標email位址詳細分類列入清單。一般來說，運行cfagent的輸出部分會被存儲在/var/cfengine/outputs中的時間戳檔中，而如果不存在，cfexecd將會建立這個子目錄。

### 1.3.4 後續

通過開始在一個單一的主機上執行這個簡單的策略，你可以通過在cfengine的控制下增加更多的主機，或者放置更多的系統配置維護方面的問題來建立你自己的cfengine執行策略。我們將會在後續章節中繼續分別討論這兩個因素。

5· 這個簡單的例子並沒有像實際情況下需要考慮至少兩個關係那樣的細緻。第一，並不是所有的非SuSE UNIX 和 Linux系統為**crontab**檔使用特定的放置位置，因此第二個類的表達需要為更多的環境考慮而變得複雜。第二，為確認對cfexecd的多次輸入並沒有記錄在最終的**crontab**檔中，編輯目錄時需要更加當心。