1．Cfengine 引言

随着科技日渐精益，对改变引入的成本将会下降。
—Alvin Toffler, Future Shock, 1970

Cfengine 是一款可自动对联网的计算机进行配置和维护的免费软件包。它适用于所有基于UNIX或类UNIX的操作系统，并且它可以通过UNIX兼容的环境/库 Cygwin 在较新版本的Windows操作系统中运行。

Cfengine 适用于管理各种环境，从一台主机到上万台主机的机群均可使用。到撰写本书为止，我们现在所知的用于一般性管理的最大安装机群约为 20，000 台。

Cfengine 可从很多方面对系统配置和维护进行管理，包括以下几点：

- 完成后期安装任务，例如配置网络界面信息。
- 编辑系统配置文件以及其它文件。
- 建立信号连接。
- 检验、更正文件许可及所有权。
- 删除无用文件。
- 压缩被选文件。
- 在网络中分发文件。
- 自动挂载NFS文件系统。
- 检查重要文件和文件系统是否存在及其完整性。
- 执行命令及脚本。
- 应用安全相关的补丁以及相似系统的修正。
- 管理系统服务器进程。

Cfengine 的目的在于执行基于策略的配置管理。从实际的应用角度来讲，这意味着 Cfengine 可以最大限度的简化系统配置及维护任务。例如：要优化一个特定系统，用户不再需要使用 Perl 或其他用户习惯的 shell 来编译一个程序来执行每项要求，取而代之的是，用户可以通过写一个更加简单的策略来描述用户希望自己的主机如何被配置。Cfengine 软件可根据这些描述来决定哪些执行方式和/或补救方法是需要被完成的。这些策略描述也可以用于确保系统的配置能如系统管理员所希望的一样被保持下来。

以下是关于这样一个策略描述带简要注解的例子：

策略范例 1： Cfengine 配置介绍

```
control:                                                    一般指示
    tmpdirs = ( tmp:scratch:scratch2 )                      定义变量列表
    actionsequence = ( files copy tidy )                    指定执行动作（按顺序的）

files:                                                      文件所有权以及保护规范
    /usr/local/bin owner=root group=bin mode=755 action=fixall recurse=1

copy:                                                       复制文件到本地系统
    solaris::                                               仅应用于Solaris系统
        /config/pam/solaris server=pammaster dest=/etc/pam.d recurse=1
    linux::                                                 仅应用于Linux系统
        /config/pam/common-auth server=pammaster
        dest=/etc/pam.d/common-auth

tidy:                                                       管理临时草稿目录
    /${tmpdirs} include=* age=7 recurse=inf
```

这个简单的配置文件被分为四个小节，每一小节又以一个冒号为结尾的关键字引入，分别是**控制：（control:），文件：（files:），复制：（copy:），以及清理：（tidy:）。**

控制（control）小节定义了一系列目录，我们将其命名为tmpdirs，这个我们将在以后中用到（在**清理（tidy:）**小节）。

文件（files）小节指定了所有在/usr/local/bin 目录里的文件都应被根用户 root 和 bin 组所有，并且定义了文件的模式为 0755（所有权者拥有所有权限，其他用户拥有读操作及执行操作）。当 Cfengine 运行时，它将修改所有与该配置文件描述不相符的属性以及权限。因此，该小节用于定义在本地二进制目录中适合可执行文件的所有权以及许可。

复制（copy）小节中描述了对 Linux 和 Solaris 系统的不同配置方法。在 Solaris 系统中，当位于主服务器上/config/pam/solaris 目录中的文件更新后，位于/etc/pam.d 目录中的文件也会随之更新。而在 Linux 系统中，只有文件/etc/pam.d/common-auth 会随着 PAM 的主配置文件的更新而更新[1]。需要注意的是：这两个规范执行着相同的底层系统配置以及维护策略：当需要的时候，相关的 PAM 配置文件将从主服务器中被更新。

1. 这样做的原因并不是非常显而易见。很多 Linux 系统使用包含文件机制的 PAM 来传播本文件的 PAM 栈数到其他所有的可用 PAM 服务的配置文件。因此，只有这个主文件会不断改变。

最后，**清洁（tidy）**小节举例说明了隐式循环的利用。例子中单一的指示应用于 tmpdirs 列表中的每一个目录文件。对于每一个目录，Cfengine 会删除在目录中所有的项，或者子目录中任意一个七天内没有被使用过的项目（包括那些以点号开始的文件名）。如同在这个示例配置文件中的其他方针，这一节执行的策略是：删除一周内未被使用的临时目录中的项目。

所有的 Cfengine 配置文件在以上所介绍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细致具体。在进一步介绍关于使用 cfengine 的更多技术细节之前，接下来将要介绍的是一些最重要的基础知识以及理论指导思想。

1.1 基本原理概念

如前所述，cfengine 在主机上运行以使他们的配置符合特定的策略。以下是这些术语的正式定义：

定义 1：主机（host）。*一般来讲主机是一台运行于操作系统（如 UNIX, Linux, Windows）之上的一台独立的电脑。有时我们也认为其是一台机器，主机也可以是如 VMware 或 Xen/Linux 环境所支持的虚拟机。*

定义 2：策略（policy）。*策略是我们希望主机以怎样的方式运行的一种规定。与其他任何计算机程序不同的是，一个策略实质上是用于描述技术细节和特征的基本文档。Cfengine 实施经过我们考虑的策略。*

定义3：配置（Configuration）。*一台主机的配置文件是它资源的实际状态，例如，文件的权限和内容，已安装软件的详细目录等等。配置文件也指某一特定的主机在指定时间的事务状态。*

我们使用cfengine的目标是什么呢？答案是，策略一致性配置。我们想要对一台或更多的主机制定一套规范，来描述他们的特性以及他们间的相互作用（或许是用来解决一个商业问题），然后我们就要把细节，执行和维护交给自动控制代理去处理：cfagent。

人类善于理解输入和思考解决办法，但是在执行方面却不是非常靠得住（*doing*）。机器和软件代理善于并可靠地执行任务，但不善于理解或找到实际的解决问题的办法。通过使用 cfengine，你可以令“人-机器”这样组织的不同部分专注于它们各自所擅长的工作。

Cfengine 在一个相对较低的水平上工作，因此这是实际上的实现方法而不是概念上的方式。不仅如此，在决定策略之时，你会发现有足够的高级概念以供参考。

1.1.1 承诺，行为和操作

一个cfengine策略可以被认为是系统对某个审计员允诺对其配置的表单。大部分承诺包含了改变主机并使其满足策略承诺的可能性。我们称这样的改变为行为或者操作。和你可能猜想到的一样，这里的审计员是cfengine它自身的一部分。如果没有满足承诺，Cfagent也是执行改变系统的技工或者外科医生。

通过以这样的方式描述它的操作，配置管理可以被认为是一种服务，该种服务与监控和维护紧密相关，并且此服务在无需将一个系统与核心权限相关联的情况下就可以依据需求而被“购买”到。

定义4：操作（Operation）。*一组改变就是一个操作。Cfengine处理对系统的改变，而操作则被嵌入到一个cfengine策略的基本文句当中。这些告诉我们策略是如何约束主机的，换句话说，就是我们怎样避免主机失控。*

以下就是一个关于文件属性的承诺的例子：

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

在该声明中有隐含的操作(动作)：在特定情况下，如果/当他们不符合这个规定时，这些操作就会改变属性。

1.1.2 收敛性

Cfengine 的一个重要特性就是收敛。这是区分它与一般的计算机语言的重要特性。该特性有助于避免系统产生分歧：即使运行在无法控制的环境下。

定义5：收敛（Convergence）。*如果一个操作总是使一台主机的配置接近于理想的、策略一致性的状态，并且该主机已经处于此种状态，那么此操作不会对主机产生任何影响，那么这个操作就是收敛的。我们可以通过以下的规则从功能型术语方面总结这一点：*

Cfengine(incorrect state) ⇧correct state
Cfengine(correct state) ->correct state

有时我们会把“正确状态”称之为“健康状态”，以此来隐喻一个配置很差的主机犹如某种疾病所带来的痛苦。

以下是用于一个 ASCII 文件编辑中的例子：

editfiles:

...

AppendIfNoSuchLine “*Important configuration line*”

该操作说明当一个文本不在某个指定文件中时，cfengine会将该给定的文本附加在该文件的末端。因此，这个策略一致性的配置是在当前文本行已经存在，或者一旦完成添加文本行到指定文件中的操作，任何过多的操作都不会发生。那么，我们认为操作**AppendIfNoSuchLine**是收敛的。

不要低估了收敛的价值。它提供了稳定性。正是由于 cfengine 的语言界面强烈阻止你做出任何“非收敛的”事情，它同样也有助于避免错误的发生。而利用该特性的代价是你必须学着以一种收敛的方式思考，对于大多数刚接触 cfengine 的人来讲这是种全新的思想。

1.1.3 类与声明：从一台主机到多台主机

使 cfengine 的策略可读的特性之一是，cfengine 有隐藏的由代理机决定所有的复杂需要的决策的能力。为达到这样的目的，cfengine 使用一种声明语言表达策略。

这种可以被声明的语言仅仅是一个结构化的句子列表(在cfengine中，它是一个策略承诺的列表)。没有任何特殊的表达次序，它只描述了一个最终的目的。达到该目的的具体方法是不明确的，是由解释规范的引擎进行评估和实现的。这与shell或Perl等细致地控制整个过程的每个步骤之类的程序性或命令性语言形成了对比。

命令性语言专注于过程。而可声明性语言则关注目的和预测的结果。

在 cfengine 中类的使用就是一个例证。类就是一种不需要很多“if-then-else”语句的决策方式。一个类就是一个标志符，当一个测试结果为真时，该布尔型变量的值就是 true。换句话说，它隐藏了一个 if 测试的结果。以适当的系统并且/或者在恰当的情况之下，一个类是用来限制 cfengine 行为的范围内。

类的好处是所有的测试都可以被隐藏于 cfengine 的之内，只有当需要或者如果需要时，结果才是可见的。

定义 6：类（classes）。*一个类是将一个或多个主机的复杂环境裁剪和限制在一个区域之内的一种方式，并可以通过一个标志或名字而被访问。类对范围有如下描述：一些事情被限制于其中的地方。*

例如，当且仅当 cfagent 运行于以 Debian GNU/Linux 为操作系统的主机上时，类 **debian** 的值是真。

1.1.4 自愿合作

每个主机保持其个体自治是 cfengine 组件的基本特性。管理员可以随时将主机从基于 cfengine 的管理中退出。该原则引出了基本的设计和执行的决定：

定义7：自治（Autonomy）。*如果信息没有明确地被其自身请求，没有cfengine组件能够接收信息。*

理解这一点很重要。这并不代表主机的集中控制不能实现。集中式控制是大多数管理员选择使用 cfengine 的方式。实际上，若想要实现集中控制，你所需要做的仅仅是为你所有的主机做出策略决定，以便于从核心权限获得策略规范。

自治并不意味着如果环境存在一些小组或有特殊需要的次文化的特殊需要，有可能是他们保持其特殊性。任何自我委派的权限都不能超越他们的本地决定。

*那么，策略来自哪里？*任何主机都依据 cfengine 可以找到本地目录（通常是类似于 UNIX 主机之上的*/var/cfengine/inputs*）的一个策略规格来工作。如果希望主机从某个核心管理器或授权获取控制，那策略必须包含启动规格说明，即“我决定，我要下载且遵守中心管理器的策略规范。”

任何时刻，每个主机都可以取消该策略决定。这是 cfengine 安全模式的核心部分。

1.1.5 可测量性

Cfengine 是按照最大的可测量性而设计的。它的可测量性至少和其他任何系统一样优秀，因为它允许最大限度的工作量的分配。

定义8：可测量的工作分配（Scalable distributed action）。*每一个主机可基于自己的本地策略副本，对检查和维护自身的操作负责。*

这不意味着你错误的决策也行得通。举例来说，当你要求一万台主机同时从同一处获取某些信息的时候，网络服务始终会成为其瓶颈。

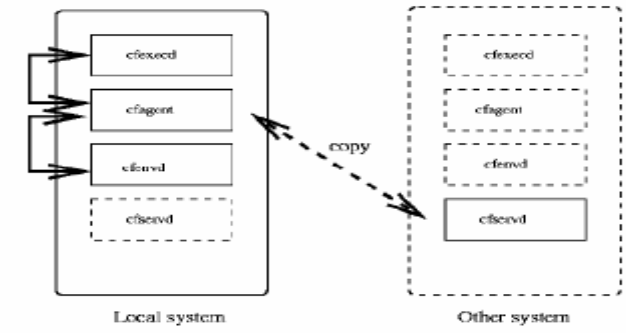
实际上每一个 Cfengine 代理保持一份本地策略的副本（不论它是在本地编写或是继承于一个核心授权），这意味着即使网络通信出现故障，Cfengine 也将持续工作。

1.2 Cfengine 的组成部分

Cfengine 软件由很多部分组成，它们是一些共同运行的独立程序。（见图表 1.1）[2]

Cfengine 的这些组成部分是：

- Cfagent: 解释策略的承诺并且以收敛的方式执行它们。代理可使用由统计监测引擎Cfenvd产生的数据，并且它能从运行于本地或远程主机上的Cfenvd中获取数据。
- Cfexecd: 执行 Cfagent，并且记录它的输出（可选择通过电子邮件寄出摘要）。它可以在一个后台程序（standalone）的模式下运行，或者可以通过 cron 在一个类似于 Unix 的系统上运行。
- Cfservd: 监控 Cfengine 的端口：提供文件数据，并在接收一个来自 cfrun 的连接的基础上启动 Cfagent。请注意，没有数据可以通过这个后台程序。
- Cfrun: 联接远程主机，并要求他们运行 cfagent。
- Cfenvd: 收集在每台主机上使用资源的统计数据，用于异常状况的检测。信息以 cfengine 类的形式被代理获得，因此代理可以及时地对异常的动态状况进行检查并作出反应。
- Cfkey: 在主机生成“公有-私有”密钥对。一般作为 Cfengine 软件安装过程中的一个步骤，你只需运行一次该程序。
- Cfshow: 一旦你对它的内部存储感兴趣，cfshow 便将 cfagent 的数据库内容以 ASCII 的格式导入。
- Cfenvgraph: 将 Cfenvd 的数据库内容导入为一种可用于图标格式显示一台主机在其环境中的一般行为。



图表 1.1 Cfengine 的组成部分以及它们之间的联系

2. 第一版与第二版之间的组成部分是不同的。因为 cfengine 版本 1 已经不再被支持，在此我们只讨论 cfengine 2，并且我们强烈建议你使用版本 2。此外，在撰写本书的同时，cfengine 的第 3 版正在完善中，但是想要它完全取代第二版还是需要很多年的时间。第三版会吸取所有基于第一与第二版的经验教训，并结合网络与系统管理研究的发展动态，逐步完善起来。

图表 1.1 举例说明了在不同主机上 cfengine 组成部分之间的关系。在一个给定的系统里，cfexecd 程序将开始运行 cfagent；它还会在 cfagent 运行中处理日志记录。此外，cfagent 会在本地系统启动运行如在主机间的文件复制的操作，并且他们以来 cfservd 系统来取得在远程系统上的数据。

1.3 开始安装

在这一节中，我们将要安装并运行 cfengine。在理解编译语言的细节之前，首先应该让 cfengine 的组成运行于一个基本、简单的策略之下，使得引擎慢慢运作起来。当你了解了它的操作方法之后，便可以一步一步建立起自己的策略。

1.3.1 建立软件

Cfengine 的安装如同其他大部分 UNIX 开源软件一样。你可以选择软件包安装版本配置到你的系统，或者从源代码自己编译。

在任何一种情况下，你都需要两个库：BerkeleyDB，用于内部数据库的使用；和OpenSSL，用于加密方法。这些库都是开放资源，并且如同cfengine一样可以免费使用。没有这些库，你将无法使用cfengine，你也不能用其他的库来替代这两个库。[3]

开始安装cfengine的第一步，需要从*http://www.cfengine.org*，或者其镜像站点的任意一处下载源代码文件。下载的包将是一个以*cfengine-2.x.x.tar.gz*命名的压缩包，其中*x.x*是指cfengine第二版的后缀版本号。

以下的几步总结了建立cfengine所需要的大体步骤：

步骤1：从开源代码安装Cfengine

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

3． cfengine所使用的数据库是基于内存的高速低级别的运行结构。他们不能用于用户数据存储。确切的说，cfengine需要一个有着很强查找能力的本地数据库。因此，SQL相关的数据库是不适用并且也不可能用于cfengine。

在类UNIX用户上安装二进制代码的默认位置是在*/usr/local/sbin*下。

这个目录有时是一个共享文件系统（例如通过NFS而安置的远程文件系统）。在网络失去连接而Cfengine一定要可以继续运行时，这将成为一个问题。为此，cfengine的目录树可以继续维持cfengine的二进制代码的复制。

1.3.2 建立你的第一个cfengine主机

终于，你可以让cfengine完成在一个新的计算机系统上安装本身的大部分工作。作为一个初学者，你应该希望学习这一项任务是如何实现的。为了实现这个目标，我们将手动安装cfengine，这样每一步骤将显而易见。

为了避免不需要的网络文件系统间的关联性，cfengine使用了同一个目录以确保其可以安装于任一主机（除了无磁盘的客户端）。其默认安装目录（通常称为“工作目录”）是*/var/cfengine*。在这里，我们可以假定cfengine的可执行文件被安装在*/usr/local/sbin*。

接下来的一步是建立cfengine工作目录树的基本结构：

步骤2：手动建立Cfengine的工作目录

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

接下来，在工作目录*bin*的子目录下（例如：*/var/cfengine/bin*）建立cfengine可执行程序的本地副本。实际运行过程中，是这些副本被执行，因此当网络在执行任务期间断掉，也不会对系统产生风险。

步骤3 复制Cfengine二进制代码到工作目录

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

现在，这些二进制代码被放在了一个可靠的位置中，接下来我们可以通过建立一个简单的cfengine策略来对这个代理进行测试。

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

策略范例2：初次测试的简单策略

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

这就是你所需要的测试cfengine的所有脚本。这个策略非常简单，其目的在于打印显示一条信息。现在可以通过运行这个代理来进行测试。在默认状态下这个代理会在工作目录下寻找*cfagent.conf*这个文件。需要注意的是我们必须在第一次运行cfagent以前运行一次cfkey命令。[4]

步骤4：运行代理以测试Cfengine的基本功能

```
# /usr/local/sbin/cfkey                    在第一次运行cfagent命令前运行一次该命令
# /var/cfengine/bin/cfagent
cfengine::/bin/echo Dange: Danger, Will Robinson!
```

现在，令人惊奇的事情发生了！在一秒的时间上立即运行cfagent命令两次，你会发现没有任何事情发生。这是正常的现象。实际上，到离你上次运行cfengine的一分钟时间内，不会有任何其他的事情发生。如果你运行cfagent –v，调用详细模式，你将会看到输出信息中包含以下内容：

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

这个信息是告诉你，cfengin觉得这样重复这个承诺动作实在太快了。我们将在后面讨论cfengine的事务处理锁时会返回到这个问题上。

现在祝贺你！你已经成功地使用cfengine了！

4． 这个命令将为本地系统建立公有-私有密钥对，并在*/var/cfengine/ppkeys*子目录下存储结果文件。它也可以在cfengine工作目录下构建*randseed*文件和其他多份附加子目录。

1.3.3 建立一个长期的配置

让cfengine运行于一个规则的基础上是件很正常的事情：每小时运行一次或者每15分钟运行一次，这完全取决于你的需求。经常性的运行cfagent不会对系统造成负担，因为引擎不会重复多余的动作（你应该还记得收敛性的特性）。我们可以通过手动编译一个*crontab*文件来完成这个目标，但是，取而代之的是，现在我们可以让cfengine来为我们完成这个目的。

编辑策略文件使之与下面的例子相匹配：

策略范例3：Cron每15分钟运行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"
         }
```

我们已经删掉了Lost in Space的策略，取代它的是根用户root的*crontab*文件。现在，这个策略可以执行一些简单文件的编辑，而不是运行一个shell命令。另外，它对以双冒号为结尾的SuSE Linux提供了一些参考提示。这是另一种cfengine类的例子，并且他们定义了后续承诺何时会被应用。

在这个例子中，我们考虑到SuSE Linux为*crontab*使用了一个不同于其他大部分操作系统的惯用性目录的事实。因此在我们的策略文件举例的后半部分，我们为SuSE用户指定了

一条规则，而对其他非SuSE用户指定了另一条规则（正如你所猜想的，“！”意味着逻辑非）。

在这两种情况下，如果在*crontab*中不存在某句话，承诺将会把这句话加入其中。[5] 我们将会在本册的后续章节中会对文件的编辑做出更多细致的解释。

修正后的策略文件也在控制小节中包含了附加的指示说明。在cfexecd开始启动的时候，它可以将由cfagent所产生的目标email地址详细分类列入清单。一般来说，运行cfagent的输出部分会被存储在*/var/cfengine/outputs*中的时间戳文件中，而如果不存在，cfexecd将会建立这个子目录。

1.3.4 后续

通过开始在一个单一的主机上执行这个简单的策略，你可以通过在cfengine的控制下增加更多的主机，或者放置更多的系统配置维护方面的问题来建立你自己的cfengine执行策略。我们将会在后续章节中继续分别讨论这两个因素。

5. 这个简单的例子并没有像实际情况下需要考虑至少两个关系那样的细致。第一，并不是所有的非SuSE UNIX 和 Linux系统为*crontab*文件使用特定的放置位置，因此第二个类的表达需要为更多的环境考虑而变得复杂。第二，为确认对cfexecd的多次输入并没有记录在最终的*crontab*文件中，编辑目录时需要更加当心。