1. Introduction

Even a site with a small network of computers is likely to have hundreds of configuration files. The content of these files determines how the overall system works and what it does; this is what makes the difference between a lab of student machines and the same set of hardware providing a Web service.

Working out what to put in the configuration files takes a lot of skill and experience. Any mistakes are likely to cause some kind of failure or security breach. A system administrator will usually be responsible for setting these files up correctly and updating them as the system and the requirements change. This is system configuration.

For a site of any significant size, this is usually too difficult or hazardous to do completely by hand. LCFG is one of several tools which have been developed specifically to help automate the configuration process.

LCFG is not for everybody. It takes a lot of commitment and effort to adopt. It requires an ongoing maintenance effort, and it works best where there is a particularly disciplined approach to site management. However, the rewards can be huge; your configurations are much more likely to be correct (and hence secure) and up-to-date. Maintaining them will take less effort, and that effort can be less skilled. Even extreme changes to the site configuration will be much easier: people often find that LCFG-managed configurations are in a constant state of flux, but this is simply because LCFG is able to respond instantly and reliably to changes in requirements.

1.1 This Booklet

This booklet is a practical guide to using LCFG for managing real-world site configurations. The rest of this chapter talks about the architecture of LCFG and explains some general principles to help you decide whether it is appropriate for your situation. If you are impatient to see what LCFG feels like in practice, you can go straight to Chapter 2 for a basic demonstration (but you will probably want to come back and read this later). The remaining chapters go into more detail about LCFG and its use:

- Chapter 2 is a “quickstart” tutorial which uses an image pre-installed with LCFG to provide a hands-on introduction.

---

1. I tend to talk about “configuration files” because these are the most common way of storing configuration information on UNIX systems—but the problems are the same even when this information is stored in databases or behind GUIs.
Chapter 3 describes the simple format used to specify configuration parameters and talks about how these parameters are manipulated by the LCFG compiler.

Chapter 4 explains the concept of LCFG components and introduces some common examples. These are the small scripts that take the configuration descriptions and turn them into specific OS configuration files.

Chapter 5 covers the practical issues of using LCFG to manage a real site.

Chapter 6 describes how to write custom LCFG components.

Chapter 7 presents a few concluding thoughts.

LCFG is still evolving, and the details of the implementation will certainly change. The main features described in this booklet are likely to remain stable, but full documentation is included with the latest software, which is available from http://www.lcfg.org. This site also contains slides and recordings of live LCFG tutorials, as well as some research papers and a pointer to the LCFG mailing list.

This booklet does not include any in-depth discussion of the system configuration principles behind the software. If you are interested in this—and it does help to explain why LCFG does things in certain ways—then you might like to read my SAGE booklet System Configuration.2

A Word About Terminology

The machine running the LCFG server code is often just called the server, but I have tried to use LCFG server whenever there is any ambiguity, to distinguish it from a machine running some arbitrary service. The term LCFG server is also used to refer to the code itself, but whether code or machine is meant should be clear from the context; similarly for the term client.

1.2 Approaches to System Configuration

Unfortunately, system configuration is still a developing subject, and there aren’t any clear standards or widely recognised approaches; most people start by gradually evolving their own scripts and tools, based on their original manual procedures.

This approach has a lot of advantages; people can gradually adopt an automated solution, learning the tools and principles as they go along. There is no “big bang,” and everyone can be reasonably confident that the system is going to continue to work in the same way as before. The sorts of tools best suited to this approach are usually based on familiar principles (e.g., UNIX shell programming)—or at least, they specify configurations in terms of familiar concepts such as processes and configuration files (e.g., Cfengine). These tools don’t involve a radical change in the way people need to think about their systems, and they don’t involve a steep learning curve.


There is a problem, however, with this incremental approach. Once the mass of configuration details has been tamed, the higher-level conceptual problems start to become more apparent: How can you smoothly manage constantly changing configurations so that services are not disrupted? How can you make sure that different people can work together on the configuration of a site without conflicting with one another? How can you be certain that your configuration is always consistent—that clients are always configured in a way that matches their servers? Can you automatically analyse your configuration to identify any unexpected interactions that might allow an attacker to compromise the system? Can you support “autonomic” recovery so that things reconfigure themselves automatically in the event of a failure?

The types of tools mentioned above are difficult to use in a way that supports this kind of higher-level thinking. For example, most people who write Cfengine (or shell) scripts to configure their firewall and their Web server don’t naturally write them in a connected way so that the firewall is automatically updated when the Web server is moved to a different machine. Other tasks, such as extracting explicit information about service dependencies, would be even harder (this is very useful if you want to automatically analyse single points of failure, for example).

Our own in-house installation consists of about 1200 machines (200 of which are servers of some kind) with very diverse configurations. For a long time, the configuration has been fully automated and “prescriptive”—we make no manual configuration changes on production machines, and we like to think of our configurations as extremely reliable and consistent. Recently, we have been interested in tools and procedures to help manage these kinds of higher-level concerns; this has spawned a few research projects, as well as a lot of practical discussion and experimentation.

LCFG\(^5\) is the configuration tool we developed to support this environment. It has evolved over 15 years and currently goes further towards addressing some of the high-level issues than any other tool that we know of. LCFG uses a very simple and consistent format for writing down configuration parameters, and the complete configuration of all the machines is usually held centrally on one LCFG server. This means that we can easily use other programs to analyse the configuration of the whole site, or even generate some aspects automatically. This is what allows us to check the consistency of configurations between clients and servers, make staged releases of configurations across the whole site, or quickly reconfigure new machines as replacement servers when one fails.

---

4. *Prescriptive* means “of, or relating to the imposition or enforcement of a rule or method.” In the past, we have sometimes mistakenly used the term *proscriptive*, which has a different meaning.

5. The name LCFG was originally derived from “Local ConFiGuration system.”
1.3 What Exactly Is LCFG?

There are three main parts to the LCFG software (see Figure 1.1):

- **The LCFG server.** This collates all of the configuration information from the source files and creates a single XML file (the profile) for each machine. The profile contains all of the configuration parameters (resources) for that machine. The profile is exported to the LCFG client using a standard Web server such as Apache.

- **The LCFG client.** This runs on every machine. It downloads a new profile for the machine whenever the configuration changes. The LCFG client works out which components are affected by the change and calls the corresponding script for each one. The client can be configured to return a simple acknowledgment to the server so that it can keep track of any systems that fail to reconfigure correctly.

- **The LCFG components.** Each machine includes a collection of scripts called components. These are responsible for translating the LCFG resources into machine-specific configuration files. Each component is responsible for a self-contained subsystem—for example, the password file or the Apache server. The component generates the configuration file and takes care of other low-level details such as restarting any associated daemons.

All of the interesting work happens in the LCFG server. There is no one-to-one correspondence between the source files and the profiles; the profile for a machine is created by assembling information from many source files according to the classes assigned to that machine. For example, a particular machine may be a “Dell GX250” running a “Web server” in “building X.” The LCFG server collates and merges all of these resources, managing any conflicts and prioritisation. This is what allows these different aspects to be managed by different people.

In addition to simply merging resources, the LCFG server can perform some more complex operations. For example, it can collate parameters from many machines and present them as part of the configuration for some other machine; this could be used to

---

6. The LCFG server sends a short notification to the LCFG client when a change occurs. The client also polls the server regularly in case it misses a notification.
collect the IP addresses of all the Web servers to include as part of the configuration for the firewall machine (this is called a *spanning map*).

The LCFG configuration is truly *declarative*. This means that it specifies what the configuration should look like rather than specifying what changes to make. The SAGE *System Configuration* booklet explains why this is a very important property. For example, it allows an LCFG client safely to miss configuration changes or receive the same changes multiple times—the configuration is simply synchronised with the most recent profile, and it is always correct.

In addition to maintaining the configuration of an existing system, LCFG supports a number of techniques for installing new systems from bare metal, preconfigured according to their LCFG specification.

### 1.4 Is LCFG for You?

Configuring a modern computing installation is complicated. Good configuration tools are like compilers in that they will help you translate your overview into a detailed implementation. But the real work lies in developing the specifications for your installation. It is not possible to take LCFG (or any other tool) out of the box and expect it to create a complete, functional site; it needs careful thought and effort to develop specifications appropriate to your needs and to express them in the appropriate language.

Adopting or changing the configuration tool for a large site is also a big commitment; people need to learn new technologies and procedures. They need to gain sufficient confidence in the system to trust it with vital services, and any new system is likely to remain in place for a long time.

Unfortunately, most sites probably develop their configuration strategies gradually, without ever making any conscious decisions based on the kind of high-level issues we have been discussing. By the time the site configuration is mature enough for these problems to become apparent, there is usually a huge investment in the existing systems.

If you are thinking about adopting a new configuration technology, you should probably spend some time evaluating the available tools against your own particular requirements. The SAGE *System Configuration* booklet provides a good background to the factors you may want to consider. Here are a few key points about LCFG:

- LCFG is designed to separate the configuration problem into two main layers. The configuration of the site is represented in a simple, uniform language. This forms a solid foundation for other processes (manual or automatic) to manipulate the whole site configuration. The components translate this into the appropriate details for particular configuration files.
- LCFG is not simply for managing desktop machines (although it is very good at this). That work often just involves slight variations on similar configurations (such as different sets of software packages). The advantages of LCFG
become more obvious when managing servers, which tend to have more diverse configurations with complex relationships among them. We manage over 200 servers running interrelated services such as DHCP, DNS, NFS, AFS, SMB, NTP, IMAP, SMTP, SSH, and HTTP. This is fairly typical for a large site.

❖ LCFG is particularly effective for larger sites where there is a mixture of skills and specialists collaborating on the management of the whole fabric. It breaks down the configuration task into at least three levels: domain specialists (e.g., Mr. Networking), who write component code and perhaps Web forms; more general system administrators, who write macros encapsulating configuration options for various services; and technicians, who assign these sets of configuration parameters to specific machines. LCFG helps these people to work together without conflicting.

❖ This LCFG model is different from the traditional one, in which individual system administrators often have responsibility for all aspects of a particular machine; our devolved approach becomes necessary as soon as configurations reach sufficiently specialisation that one person can no longer be expected to understand all of the implications.

❖ LCFG ships with a large number of standard components for common subsystems such as Apache, DNS, and Kerberos. These components translate the LCFG parameters into the corresponding configuration files. This means that a lot of things can be configured simply by providing parameters to standard components. Almost every site will have some special requirements, though, and some editing or creation of local components will almost certainly be necessary. The amount of work is usually small, but it does require coding ability, and it takes time to understand the framework.

❖ The components are written within a simple, standard framework. This means that they can be shared between sites much more easily than arbitrary scripts.

❖ LCFG can be used as a prescriptive tool, i.e., to manage every aspect of a machine so that there is no manual intervention (or configuration by other tools). There are lots of advantages to doing this, and the SAGE configuration booklet covers the topic in some depth. But this does not have to be the case—LCFG can easily be used to manage just a few configuration files. The rest of the system can be managed manually or configured using other tools.

❖ What is important is to avoid a conflict between these approaches. If LCFG is being used to manage some aspect of the system, administrators must trust the tool and not fight it by making manual changes. This sometimes requires discipline and education.

❖ LCFG configuration parameters are specified in a uniform format so that they can easily be interpreted and generated. These parameters are translated (by components) into machine-specific configuration files. In one sense, the LCFG specification is higher-level: it represents more information than the configuration files themselves. This means that it is not really practical to take an exist-
ing system and automatically generate an LCFG configuration for it. To do this, you need to understand the configuration of the machine (the “why” as well as the “what”) and manually create an equivalent LCFG configuration.

❖ A tool which interfaces to so many system components is naturally sensitive to updates and variations between OS versions. Our own site is based largely on Red Hat/Fedora/Scientific Linux, and these are well supported; tested versions of many components are available for various releases of these operating systems. With a good understanding of the framework, porting the core and relevant components to other UNIX-based systems should be fairly straightforward; we are running production systems based on Solaris and Mac OS X, for example. But these are not so well supported or packaged for export, and more commitment would be required to manage these platforms.

❖ There is no technical reason why LCFG should not be used under other operating systems A demonstration LCFG client has been created for Microsoft Windows, together with components for Windows-specific applications such as Registry modification. However, Microsoft and the Windows world have their own approaches to configuration, and we have never had strong enough need to develop this platform.7

1.5 The LCFG Software

A lot of software is available on the LCFG Web site, which can be confusing. This is largely because the Web site includes all of the components that we have developed for various subsystems. You will probably find an existing component for many of the things you want to manage. The software is continually evolving, and we continue to export all of the regular releases. This means that you can continue to download software compatible with the version you are currently running without being forced to upgrade. Finally, LCFG components are often tied closely to the version of the operating system, and we export packages and tested versions for several different operating systems.

The packages are available individually or in bundles:

❖ The “core” bundle contains the libraries and shared code, the LCFG server and client, and a few basic components. This is a reasonable starting point for experimenting with the system (but see Chapter 2 for the easiest way to get started).

❖ The “standard” bundle contains the most common components.

❖ The “optional” and “contributed” bundles contain additional components.

There are also sets of prerequisite packages (e.g., Perl modules) which are not part of LCFG and not normally shipped as part of the OS. Various disk images (e.g., a demonstration system as a VM image) and sets of sample configuration files are also available.

The pre-packaged software is shipped in the native format for the OS, usually RPM. Direct access to the CVS repository is available for building on other platforms.

7. In particular, we run very few Windows servers.