Large Scale Linux Configuration with LCFG: An Update

by Alexander Holt <lex@fixedpoint.org>

DICE Computing Environment Project
Division of Informatics
University of Edinburgh

1 Introduction

This note updates 'Large Scale Linux Configuration with LCFG'[1] by Paul Anderson and Alastair Scobie, which appeared in the proceedings of the Atlanta Linux Showcase, 2000. Here, we describe how LCFG has recently evolved from the description given there.

2 Summary

Significant changes to the architecture of the LCFG system include:

- Each per-machine configuration is now represented as an XML document, the *node profile*.
- Machine configurations are distributed by HTTP, rather than as a NIS map.
- Control scripts have the opportunity to reconfigure the system as soon as one of their configuration resources changes.

These modifications have been implemented via LCFG *adaptors*,[2] so that existing source files and control scripts can benefit from the new developments without themselves requiring changes.[3]

Some experimental features are:

- Contexts: a constrained mechanism for varying a client's configuration without changing the source files.
- Acknowledgements: regular notification from clients to configuration server about their current configuration state.

[1] `http://www.dcs.ed.ac.uk/home/paul/publications/`

[2] Distributed in the `lcfg-adapt` RPM.

[3] There are some minor exceptions.

A further innovation is provision for installing PXE-capable machines over the network, without the use of a boot floppy or CDROM.

3 Node profiles

Formerly, all per-machine configuration resources were held in a single NIS map. There is a new format for this information, based on an XML document type, called the node profile. Some of the advantages of this change are:

- Because there is a DTD, there is a notion of validity for the node profile (which can be validated on a per-machine basis).
- Both resource keys and resource values may now have a well-defined structure. In particular, compound values—such as lists—can be given a more natural treatment.
- The list of RPMs for a machine is part of the node profile, and thus integrated with all other configuration information.
- There is a syntax which allows one resource to make an explicit reference to the value of another resource.

Another consequence is that all valid resources must now have values. Previously, a resource could be 'undefined', which the control script was able to detect and substitute with some default value. (This is one of the changes which can occasionally require old scripts to be modified. Some meta-information also needs to be added to the source files, to enable the adaptors to construct appropriate XML, for example in the case of list values.)

The `mkxprof` program runs on the configuration server and is responsible for generating XML node profiles from the configuration source files.

A future phase of LCFG development will consider an improved format for source files, but we intend the XML node profile specification to remain relatively stable from now onwards.

4 Profile distribution

Each client machine now fetches its own node profile using HTTP to the configuration server, rather than via NIS. The program `rdxprof` runs on the client (normally as a daemon), fetches the profile, and caches it locally. The caching function is scheduled to be replaced by a dedicated configuration cache manager (CCM) program.

A third adaptor program, `ldxprof`, provides a transparent compatibility layer for existing control scripts, enabling them to load resources in the usual way. Ultimately, 'new generation' control scripts will use a different interface to the local configuration cache that takes advantage of its richer structure.

5 Reconfiguration on changes

It is now possible for a specified method of a control script to be called whenever any of that script's resources have changed. This encourages control scripts to move away from reading resources at boot time (and caching them), towards relying on the local configuration cache and update mechanism for appropriate 'pings' when reconfiguration may actually be required.

6 Contexts

Until now, each per-machine configuration was entirely determined by the contents of the global source files. This achieves maximum central control, and has clear advantages. There are a number of situation, however, when it seems beneficial to allow a machine's configuration to vary in limited ways, without requiring changes to the source files. Examples include:

- Network schemes for portables. The network connectivity of a portable can vary rapidly over time; it may be disconnected, on a wireless connection, on a fixed ethernet, connected via dialup PPP, etc. When the network scheme changes, so should a collection of configuration resources.
- Debugging levels. It can be useful to dynamically control diagnostic output, or other debugging behaviours.
- Switching between 'production' and 'maintenance' states.

We call a configuration variable which can change at run time in this way a *context*.

Both the node profile and the current LCFG adaptors allow configuration resources to depend on contexts. It is important to realize that the control scripts themselves are unaware of context changes, except in so far as a context change happens to modify one of their resources, in which case they are notified of the change in the usual way (as described above).

7 Acknowledgements

It is now possible for each client to send regular acknowledgements to the configuration server, including information such as its current IP address (which may vary for portables), and the version of node profile it is actually using (which can be used to tell which machines are currently up to date).