Augeas: A Configuration API

Raphaël Pinson

June 1, 2012

Augeas: A configuration API

This book would not be what it is without the help of many. In particular, I wish to thank David Lutterkort for writing Augeas and providing great insights about Augeas internals. Doing proper typesetting with LaTeX can be tricky, and Damien Wyart deserves a big thanks for providing his expertise on the subject.

Contents

Introduction

Augeias was king of Elis,
in some accounts a son of
Helios,
in others of Poseidon,
and in still others of Phorbas.

The Library 2.88
APOLLODORUS

In the world of Unix systems, there is no standard way to store configuration. Countless formats can be found, from simple shell variables files to complex, specific, multi-level formats, making the infamous /etc directory a sort of digital Augean stable.

Augeas provides a way to cleanly manage these configuration files through a unified API.

Configuration Data Editing Approaches

While system administrators are well aware of the heterogenous state of the configuration data on Unix systems, these configurations have to be edited automatically in many situations.

There are three main approaches to the issue of automating configuration data editing on Unix systems.

Configuration file	Format
`/etc/default/*`	shell variables
`/etc/fstab` `/etc/mtab`	fstab format
`/etc/hosts`	hosts format
`/etc/passwd` `/etc/shadow`	passwd format

`php.ini`
`my.cnf`
`gdm.conf`
`puppet.conf`
	INI file

`ntp.conf`	NTP format

Some common configuration files and their format

Keyhole Approaches

While most programming languages provide modules to edit at least the most common formats, a lot of system administrators and developers have had to manipulate these files using string editing tools such as sed, awk or cut, or even to write scripts dedicated to a specific parsing job. In the majority of these cases, the results are not guaranteed, and you are likely to ruin the configuration files if your parsing expressions are wrong or the file layout changes.

Configuration management tools such as Cfengine provide tools (like AppendIfNoSuchLine) to achieve keyhole approaches, but the problems are similar to using string editing tools: you have no guarantee that the result will be a valid configuration file, and you have to write the regexps yourself.

Augeas is particularly useful to ease and secure this kind of approach.

Greenfield approaches

When you are the main system administrator of a machine and you wish to control all the parameters of the machine, you may want to provide the configuration files entirely. In this case, it is common to set a repository of configuration files, or a database, which will contain the whole configuration as will be deployed to the machines.

Templating

If you wish to control whole configuration files but you need a fine-grained mechanism to generate these files, templating is probably the best approach. There are lots of options to achieve this. Puppet, for example, provides ERB templates that let you easily generate configuration files from exported variables.

A Unified Configuration API

In many cases, system administrators and users want to change a single value in their configuration without affecting the rest of it. This is often achieved using the keyhole approach, which as we have seen is not very reliable. A better approach would be to have a unified configuration API that lets you modify configurations in a simple and reliable way, ensuring that the modified files are valid configuration files. This is the goal of Augeas.

What Augeas is not

A principle on Unix systems ensures the stability and simplicity of the system tools: each tool attempts to do one thing, and to do it well. Augeas is no exception to this rule, so that Augeas is as much defined by the things it does not try to accomplish as by its goals.

Before we dive into what Augeas can do for you, it is important to note the following points.

Not an abstraction layer

Augeas does not attempt to provide an abstraction layer from the native configuration format. The organization of the Augeas tree mirrors closely that of the configuration files it represents.

As an example, if the `/etc/foo.conf` configuration file contains an include statement such as the following:

```
#include /etc/foo.d/*
```

Augeas will not attempt to parse the contents of the files in `/etc/foo.d/*` and add them to the `/etc/foo.conf` tree. Instead, it will provide a tree like the following:

```
/files/etc/foo.conf
/files/etc/foo.conf/#include = /etc/foo.d/*
```

`#include` is just a parameter of the `/etc/foo.conf` configuration file and `/etc/foo.d/*` is the value of the parameter. The contents of `/etc/foo.d/*` will probably appear in the tree if the lens is able to parse them, but in no way will Augeas make a logical link between `/etc/foo.conf` and `/etc/foo.d/*`.

Other software provide this kind of abstraction layer. This is the case of Config::Model, which can use Augeas as a backend, and is able to understand the logic of a configuration files, such as include statements, or the link between several statements in a configuration file.

Another consequence of this non-goal is that the statements in the Augeas tree will appear in the same order as they do in the configuration file. In some cases, it is technically possible to write Augeas lenses that invert parameters or otherwise modify the logic of the configuration statements. Doing this is not recommended, as the Augeas tree should stay as close as possible in its logic to the configuration files it is representing to provide maximum flexibility.

Not a cross-platform abstraction layer

For a similar reason, Augeas does not attempt to be a cross-platform abstraction layer. When Augeas finds Apache configuration files in `/etc/httpd/httpd.conf` on some operating systems and in `/etc/apache2/apache2.conf` in others, these files will be represented in the tree as `/files/etc/httpd/httpd.conf` and `/files/etc/apache2/apache2.conf` respectively.

Similarly, some operating systems provide their network configuration in `/etc/sysconfig/network` while others use `/etc/network/interfaces`. Augeas will represent these two files in different parts of the tree, and the tree will mirror the way each of these files is organized, without attempting to provide a unified way to configure network interfaces across these operating systems.

Other projects such as netcf[1], based on Augeas, provide a cross-platform abstraction layer to manage network interfaces regardless of the operating system, but it is not Augeas' goal.

No remote management support

When you are dealing with a whole fleet of servers and wish to set a parameter for each of them, it is useful to use a tool that has a

[1] `https://fedorahosted.org/netcf`

network protocol for remote management. Augeas does not attempt to be that tool, and the Augeas API is designed to be a local API.

Remote access to the Augeas API are meant to be added on top of it, not in it.

Puppet[2] is an example of configuration management tool which supports Augeas as a native type and provides remote management functionality.

Very little modelling

The goal of Augeas is not to understand or otherwise interpret configuration files. As stated before, Augeas does not attempt to provide an abstraction layer, but it provides a light modelling, although very close to the organization of the configuration files.

For example, an `/etc/hosts` line like the following:

```
192.168.0.10    aslan   # Added by NetworkManager
```

will be represented by the following tree:

```
/files/etc/hosts
/files/etc/hosts/1
/files/etc/hosts/1/ipaddr = ”192.168.0.10”
/files/etc/hosts/1/canonical = ”aslan”
/files/etc/hosts/1/#comment = ”Added by NetworkManager”
```

The order of the statements is strictly kept; Augeas does not interpret the configuration files per se, but it labels each of the fields on the line to ease access to individual configuration items.

[2]`http://www.puppetlabs.com`

Installing Augeas

Installing from source

You might want to install Augeas from source if your distribution does not provide any binary packages, if you simply want to use the latest version of Augeas, or tune compilation parameters.

You can find the latest source code on the Augeas website:

`http://augeas.net/download/`

Next, install the necessary dependencies to build Augeas. The minimal dependencies you will need are the readline headers. You can use one of these commands to install them:

```
$ sudo yum install readline-devel
$ sudo apt-get install libreadline-dev
```

Then, extract, compile and install:

```
$ tar xvzf augeas-0.8.0.tar.gz
$ cd augeas-0.8.0
$ ./configure
$ make && sudo make install
```

Installing from binary packages

Most distributions provide Augeas packages, often split up into the shared library, the lenses provided with Augeas and the command-line tools.

On Red Hat or Fedora derivatives, you can install the augeas package using yum:

```
$ sudo yum install augeas
```

Or on Debian and Ubuntu systems, you can install the Augeas library and the augtool command-line interface with the following:

```
$ sudo apt-get install augeas-tool
```

You might also want to install the documentation package with:

```
$ sudo apt-get install augeas-doc
```

Installing from the development head

Augeas' code is maintained in a public repository which can be cloned and used to test the latest features and fixes before they are released.

If you wish to build and install from the development head, you will need `git` [3], `autoconf`, `automake` and `libtool`, as well as the normal dependencies to build Augeas from source. Then follow these instructions:

```
$ git clone git://git.fedorahosted.org/augeas.git
$ ./autogen.sh
$ ./configure
$ make && sudo make install
```

Building documentation

Augeas provides documentation in the form of LaTeX files and inline documentation in the C API source and lenses, formatted in the NaturalDocs[4] format.

[3] provided by the `git-core` package on Debian-based distributions

[4] `http://naturaldocs.org`

If you want to build this documentation, begin with the instructions in section 'Installing from source' on page xvii to retrieve the source code.

On Ubuntu, you can find this documentation already built in the 'augeas-doc' package[5].

Building the PDF documentation

In order to build the PDF documentation, you will need to install `pdflatex` on your system, with one of the following commands:

```
$ sudo yum install texlive-latex
$ sudo apt-get install texlive-latex-base
```

Then add the `--with-pdfdocs` flag to the `./configure` call and call `make`:

```
$ ./configure --with-pdfdocs
$ make
```

This will generate PDF files in the `doc` directory, which contain mostly theorical information on lenses and their implementation in AugeasSee *Bidirectional transformations* on page 11.

Building the NaturalDocs documentation

The NaturalDocs documentation covers the inline documentation for the C API and the lenses shipped with Augeas.

[5] See *Installing from binary packages* on page xvii.

The NaturalDocs documentation is available online at `http://augeas.net/docs/references/lenses`

In order to build this documentation, you need to install `naturaldocs` on your system, using one of these commands:

```
$ sudo yum install NaturalDocs
$ sudo apt-get install naturaldocs
```

You can then build the documentation by calling `./configure` with the `--with-naturaldocs=HTML` or `--with-naturaldocs=FramedHTML` flags. The first flag will include the HTML header in each generated file, while the second flag will generate HTML frames to read the documentation:

```
$ ./configure --with-naturaldocs=HTML
$ make
```

This will produce HTML files in the `doc/naturaldocs/output` directory. The `c_api` directory will contain the generated documentation for the C API. The `lenses` directory will be the documentation for the lenses provided with Augeas. You can open the `index.html` file of one of these directories to access the full documentation.

Conventions

This book uses the following conventions:

- Filesystem paths, Augeas calls and Unix commands are written in a monospace font;
- When lines are too long in an output, an antislash (\) is added and the rest of the line is reported to the next line with an indentation.

1 Exploring augtool

King Augeas' fleecy flocks, good Sir,
feed not all of one pasture nor all upon one spot,
but some of them be tended along Heilisson,
others beside divine Alpheüs' sacred stream,
others again by the fair vineyards of Buprasium,
and yet others, look you, hereabout;
and each flock hath his several fold builded.

Idyll 25.7
THEOCRITUS

Augeas is primarily a C library with bindings but it also provides a command-line tool called `augtool`, which we will be using in the following examples. In chapter 4, we will see how to use the API and bindings directly.

Parsing your System Configuration Files

The first thing we might want to do is to see how Augeas sees your system configuration files. Fire up `augtool`:

```
$ augtool
```

This will give you an interactive shell which passes commands to Augeas. Augeas transforms your configuration files into a tree, which has two nodes at its root: `/augeas` and `/files`. The `/augeas` node contains metadata, which we will be looking at in chapter 5, while `/files` contains the representation of the files Augeas was able to parse. You can see these two nodes by typing `ls /`:

```
augtool> ls /
augeas/ = (none)
files/ = (none)
```

What does that mean? We see the two nodes at the top of the Augeas tree, and we see that neither of them has a value. In the Augeas tree, each node can have children and a value associated with it.

`ls` is an `augtool` command which lists the children of the given node and gives their value (if any).

You can see which files (or directories containing files) were successfully parsed by Augeas in `/etc` by typing `ls /files/etc`:

```
augtool> ls /files/etc
nsswitch.conf/ = (none)
odbc.ini = (none)
passwd/ = (none)
ntp.conf/ = (none)
services/ = (none)
sysctl.conf/ = (none)
shells/ = (none)
samba/ = (none)
securetty/ = (none)
crypttab/ = (none)
...
```

Let’s inspect the contents of the first line of `/etc/fstab` in the Augeas tree. We can use the `print` command to inspect nodes and their values recursively:

```
augtool> print /files/etc/fstab/1
/files/etc/fstab/1
/files/etc/fstab/1/spec = ”proc”
/files/etc/fstab/1/file = ”/proc”
/files/etc/fstab/1/vfstype = ”proc”
/files/etc/fstab/1/opt[1] = ”nodev”
/files/etc/fstab/1/opt[2] = ”noexec”
/files/etc/fstab/1/opt[3] = ”nosuid”
/files/etc/fstab/1/dump = ”0”
/files/etc/fstab/1/passno = ”0”
```

Each of the child nodes beneath the `1` node refers to a part of a single line in the `/etc/fstab` file. The filesystem options are further split into separate nodes under the `opt` node so they can be managed individually.

What if we only wanted to find the `opt` nodes of this first line? The `match` command lets us find the nodes matching an expression:

```
augtool> match /files/etc/fstab/1/opt
/files/etc/fstab/1/opt[1] = nodev
/files/etc/fstab/1/opt[2] = noexec
/files/etc/fstab/1/opt[3] = nosuid
```

Now, we might want to get the value of the single node matching an expression, and make sure that this node is unique. For example, if we want the value of the first `opt` node of this first line, we could use the `get` command:

```
augtool> get /files/etc/fstab/1/opt[1]
/files/etc/fstab/1/opt[1] = nodev
```

To leave the `augtool` session, you can type `quit` or `^D`:

```
augtool> quit
```

Using a Fakeroot

It is often useful to play with `augtool` when you want to understand the Augeas tree or try XPath expressions. However, you likely don't want to play with the files in your `/etc` directory and take the risk of ruining your system. Augeas lets you set a fakeroot so that the files parsed and modified by Augeas are taken from this root instead of the `/` directory of your system.

In `augtool` you can set this fakeroot by using the `--root` (or `-r`) option:

```
$ mkdir -p myroot/etc
$ rsync -av /etc/ myroot/etc
$ augtool -r myroot
```

In general, you can also set the location of this fakeroot with the `AUGEAS_ROOT` environment variable:

```
$ export AUGEAS_ROOT="$(pwd)/myroot"
$ augtool
```

This option can also let you modify files inside a chroot for example.

Modifying Files

We have seen already how Augeas lets you parse your configuration files in a unified way. The Augeas tree is not only a parsing facility as Augeas exposes commands to modify the tree and save the changes to the original files.

The fakeroot option will be useful for us here, in order to modify the files without affecting the system. We will also use the `--backup` option in `augtool` so that the original files are preserved with a `.augsave` extension.

```
$ augtool --backup --root myroot
augtool> rm /files/etc/fstab/1/opt[3]
rm : /files/etc/fstab/1/opt[3] 1
augtool> print /files/etc/fstab/1
/files/etc/fstab/1 /files/etc/fstab/1/spec = "proc"
/files/etc/fstab/1/file = "/proc"
/files/etc/fstab/1/vfstype = "proc"
/files/etc/fstab/1/opt[1] = "nodev"
/files/etc/fstab/1/opt[2] = "noexec"
/files/etc/fstab/1/dump = "0"
/files/etc/fstab/1/passno = "0"
augtool> save
Saved 1 file(s)
augtool> quit
$ diff -u myroot/etc/fstab myroot/etc/fstab.augsave
--- myroot/etc/fstab    2011-03-14 23:46:07.000000000 +0100
+++ myroot/etc/fstab.augsave    2010-09-30 08:45:53.000000000 +0200
@@ -5,7 +5,7 @@
 # devices that works even if disks are added and removed. See fstab(5).

 # <file system> <mount point>   <type>  <options>       <dump>  <pass>
-proc            /proc           proc    nodev,noexec 0       0
+proc            /proc           proc    nodev,noexec,nosuid 0       0
 /dev/sdb1 /               ext4    errors=remount-ro 0       1
```

Listing 1.1: Removing an option in fstab

In listing 1.1 on page 5 we change the filesystem options specified on the first line of `/etc/fstab` by removing the third `opt` node. The `rm` command on line 2 removes only the `opt` node we specified. Line 3 tells us that the `rm` command removed only one node, the `/files/etc/fstab/1/opt[3]` node. Lines 4 through 11 show us the `/files/etc/fstab/1` tree without the removed node.

On line 12, we call the `save` command. This command tells Augeas to save the tree back to the configuration files. Augeas inspects the files and tries to apply the new tree to them. In our case, the `save` command was successful as line 13 tells us, and one file was modified, which is what we expected. We can then quit the `augtool` session by typing `quit` on line 14. We can then quit the `augtool` session by typing `quit` on line 14.

We use the `diff -u` command on line 15 to inspect the changes made by Augeas to the file. As expected, only the first line that is not empty or a comment was modified. Lines 22 and 23 in the listing show us the differences between the old and new lines. We can see that only the third option has been removed, and that the spaces have been strictly preserved. The rest of the file was left untouched.

Preserving existing files

Augeas offers two options to preserve the existing files when saving the tree. In `augtool`, these options can be triggered with the following flags:

- `--backup` will save the original file with the extension .augsave and write the new file under the original file name;
- `--new` will save the modified file with a .augnew extension and leave the original file untouched.

These options actually modify the value of the `/augeas/save` node in the Augeas tree[1].

Locating nodes in files

The span metadata were added in Augeas 0.8.0. For performance reasons, they are not activated by default. This functionality can be activated by the `AUG_ENABLE_SPAN` flag or the `--span` flag in `augtool`.

You can see if the `span` functionality is activated in the current session by looking at the `/augeas/span` node[2]:

```
augtool> get /augeas/span
/augeas/span = enable
```

The data are then available via the `span` command in `augtool`.

```
1  $ augtool --span
2  augtool> get /files/etc/ntp.conf/driftfile
3  /files/etc/ntp.conf/driftfile = /var/lib/ntp/ntp.drift
4  augtool> span /files/etc/ntp.conf/driftfile
5  /etc/ntp.conf label=(67:76) value=(77:99) span=(67,100)
6  augtool> quit
7  $ head -c100 /etc/ntp.conf  | tail -c+67
8  
9  driftfile /var/lib/ntp/ntp.drift
```

Listing 1.2: Getting the position of a node with span

Line 5 in listing 1.2 on page 7 indicates that:

- The `driftfile` label was found in the file between positions 67 and 76. This also means that `driftfile` is a dynamic key,

[1]See *The save node* on page 22.
[2]See *The span node* on page 28.

not a static label[3];

- The value of the `driftfile` node was found between positions 77 and 99 in the file;
- The whole span of the node is between positions 67 and 100 in the file. The span is one character further than the value, since the `\n` character is considered part of the lens matching the node, but is excluded from the value.

We verify on line 9 that the data located between positions 67 and 100 in the file correspond to the `driftfile` key and the value returned by the `get` command on line 3.

Scripting with augtool

In addition to running as an interactive shell, `augtool` can take commands from the command line or `stdin`:

```
$ augtool ls /files
etc/ = (none)
$ echo "ls /files/" | augtool
etc/ = (none)
```

This allows to write shell scripts that send commands to `augtool`.

The `--autosave` option in `augtool` allows you to ommit the save command.

Listing 1.3 on page 9 shows an example of a bash script wrapping around `augtool`. Lines 2 through 5 define the wrapping function `do_augtool` which is then called on line 7. Commands are separated with `\n` so they get passed line by line to `augtool` through `echo -e`.

[3] It was declared with the `key` keyword, not with `label`. See chapter 7 on page 33

```
#!/bin/bash
function do_augtool() {
   local command="$1"
   echo -e "$command" | augtool
}

do_augtool "set /files/etc/hosts/1/canonical alice\nsave"
```

Listing 1.3: Piping commands to augtool in a bash script

Using augtool as an interpreter

`augtool` can also take commands from a file:

```
$ cat commands.augtool
ls "/files"
$ augtool --file commands.augtool
etc/ = (none)
```

Listing 1.4: `augtool` takes a command file as argument

This allows to use `augtool` as a script interpreter in a shebang and write self-executable `augtool` scripts (using the `-f` short version of the option):

```
$ cat commands.augtool
#!/usr/bin/augtool -f
ls "/files"
$ chmod +x commands.augtool
$ ./commands.augtool
etc/ = (none)
```

Listing 1.5: Using `augtool` as an interpreter

Dropping into an interactive session

When `augtool` takes commands from the command line, `stdin` or a file, it doesn't start an interactive session. If you wish to pass commands to `augtool` for preprocessing and run an interactive command afterwards, you can use the `--interactive` flag:

```
$ echo "set /files/etc/hosts/1/canonical alice" | augtool --interactive
augtool> get /files/etc/hosts/1/canonical
/files/etc/hosts/1/canonical = alice
```

Listing 1.6: Setting a single value in `augtool`

The `--interactive` option only works for `stdin` and file input.

This option also allows you to make scripts that set up an environment and drop you in an interactive shell:

```
$ cat shell.augtool
#!/usr/bin/augtool -if
set /files/etc/hosts/1/canonical alice
$ chmod +x shell.augtool
$ ./shell.augtool
augtool> get /files/etc/hosts/1/canonical
/files/etc/hosts/1/canonical = alice
augtool> quit
```

Only concatenated short options can be used in shebangs, hence the use of `-if`.

2 Bidirectional transformations

Augeas uses files called lenses, written in a specific language for Augeas, which is similar to OCaml. Lenses are programs that are said to be bidirectional.

The Need for Bidirectional Transformations

Traditional programs take data as input and produce data as output, but cannot use the same code to go from the output data back to the input data. In other words, traditional programs are not bidirectional, they work only in one direction.

For example, the following transformation:

```
$ echo ”I have food in my fridge” | sed -e ”s/foo/bar/g”
I have bard in my fridge
```

transforms all `foo` occurrence into `bar` in the string, but it cannot go back from a `bar` string to a `foo` string.

The need to transform from one format and back is quite common, and has traditionally been addressed by writing two programs, one for each direction of the transformation.

In recent years, the Harmony Project[1] has been working on the mathematical conditions for programs to be bidirectional, or even bijective. They came up with a language called Boomerang, which implements their theory.

Other projects such as biXid[2] or XSugar[3] have also been working on this same goal concurrently.

XSugar provides a way to transform between XML and non-XML data models, while biXid allows transformations between two XML data models.

A Bit of Theory

What is a bidirectional transormation

Lenses

Identity and conditions of bidirectionality

Bidirectional transformations in Augeas

Augeas lenses

Lenses typechecking

The case of recursive lenses

[1]`http://www.seas.upenn.edu/~harmony/`

[2]`http://arbre.is.s.u-tokyo.ac.jp/~hahosoya/papers/bixid.pdf`

[3]`http://www.brics.dk/xsugar/`

3 Path Expressions

Augeas maps configuration files into a tree, and lets you access this tree using XPath expressions. In this chapter, we will inspect the various XPath expressions offered by Augeas, and give examples of what you can achieve with them.

Generalities on XPath expressions

XPath expressions are an XML parsing and modifying facility.

Using globs

When you write XPath expressions, you might want to match generic nodes or nodes at any level of the tree. There are two operators for that:

- `*` as a node name matches any node;
- `//` matches on any sublevel of the tree.

Examples:

```
/files/etc/hosts/*
```

will match all children nodes of the `/files/etc/hosts` node.

```
/files/etc/hosts//canonical
```

will match all `canonical` nodes under the `/files/etc/hosts` node, at any sublevel.

```
/augeas//error
```

will match all `error` nodes at any sublevel under the `/augeas` node.

Conditionals

Filtering on node names is often not enough to find what you want. You will often wish to find nodes defined by their value or subnodes. XPath offers a syntax of conditionals using square brackets.

Examples:

```
/files/etc/hosts/*[canonical = ”alice”]
```

will match the children nodes of `/files/etc/hosts` that have a `canonical` subnode with value `alice`.

```
/files/etc/hosts/*/canonical[. = ”alice”]
```

will match `canonical` nodes two levels under the `/files/etc/hosts` node that have value `alice`.

In contrast to most XML trees, the Augeas tree contains no attributes, but only nodes with values and children. For this reason, it doesn't use conditional syntaxes featuring the `@` prefix, which is common to many standard XPath queries.

Conditionals can be combined. See these examples:

```
/files/etc/hosts/*[ipaddr = ”127.0.0.1”][canonical = ”alice”]
```

will match the children nodes of `/files/etc/hosts` that have both a `ipaddr` sudnode with value `127.0.0.1` and a `canonical` subnode with value `alice`.

Union of paths

You can use `|` to achieve the union of two paths:

```
augtool> match '/files/etc/fstab | /files/etc/hosts'
```

will return the nodes matching `/files/etc/fstab` as well as the ones matching `/files/etc/hosts`.

Functions

To enrich the filtering you can achieve with conditionals, Augeas provides a set of functions which can be used in conditional context.

The last() function

The position() function

The label() function

The count() function

The regexp() function

Node references

In addition to functions, it is often necessary to refer to nodes relatively as you build complex XPath expressions. Augeas provides special node references for that.

Using variables in paths

Augeas provides two ways to declare variables.

defvar

defnode

Using variables to express conditionals

Ensuring idempotence

```
augtool> set ’/files/etc/php.ini/PHP/extension[. = ”foo.so”]’ foo.so
```

4 Using the C API and Bindings

He cleaned in one day the ox
dung of King Augeas,
Jove helping him for the most
part.
By letting in a river
he washed away all the dung.

Fabula 30
HYGINUS

So far, our examples have been done using `augtool`, the CLI interface to Augeas. However, Augeas is first and foremost a C library.

Using the C API

API Flags

Using Bindings

Haskell bindings

Java bindings

Perl Bindings

PHP bindings

Python Bindings

Installation

Initialization

Synopsis:

```
    def __init__(self, root=None, loadpath=None, flags=NONE)
```

Initialize the library.

Use `root` as the filesystem root. If `root` is None, use the value of the environment variable AUGEAS_ROOT. If that doesn't exist either, use `/`.

`loadpath` is a colon-spearated list of directories that modules should be searched in. This is in addition to the standard load path and the directories in `AUGEAS_LENS_LIB`.

`flags` is a bitmask made up of values from `AUG_FLAGS`.

Example:

```
import augeas
a = augeas.Augeas(root="fakeroot")
```

The get method

Synopsis:

```
def get(self, path)
```

Lookup the value associated with `path`. Returns the value at the path specified. It is an error if more than one node matches `path`.

Example:

```
val = a.get("/files/etc/ftab/1/canonical")
```

Ruby Bindings

5 Augeas metadata

We have seen earlier that the `/augeas` top node exposes Augeas metadata which can be parsed and modified in the same fashion as the `/files` data. This chapter will focus on documenting the various parts of the `/augeas` tree and their functions.

The root node

The `/augeas/root` node contains the root of the Augeas tree. This is the variable which can be set via either the `AUGEAS_ROOT` environment variable or the `--root` option to `augtool`.

Example:

```
$ augtool --root fakeroot
augtool> print /augeas/root
/augeas/root = "fakeroot/"
```

Listing 5.1: Inspecting /augeas/root

As of Augeas 0.8.0, this node is purely informative: changing its value has no effect on the way Augeas works.

The version tree

/augeas/version is a tree which contains several informations:

- The top node has the version of Augeas as its value;
- The save node contains mode nodes which list the known saving modes for this version of Augeas;
- The defvar node contains **what exactly??**.

```
augtool> print /augeas/version/
/augeas/version = "0.8.0"
/augeas/version/save
/augeas/version/save/mode[1] = "backup"
/augeas/version/save/mode[2] = "newfile"
/augeas/version/save/mode[3] = "noop"
/augeas/version/save/mode[4] = "overwrite"
/augeas/version/defvar
/augeas/version/defvar/expr
```

Listing 5.2: Inspecting /augeas/version

The save node

The /augeas/save node contains the saving mode used by Augeas for the session. The value of this node must be one of the values listed in the /augeas/version/save/mode nodes.

If this node is modified during the session, it will affect the behaviour of the save call whenever it is executed.

The load tree

The `/augeas/load` tree contains the lenses metadata. For each lens loaded in the Augeas session, it lists 3 types of nodes:

- a `lens` node, which specifies the name of the module used by this lens;
- `incl` nodes for each inclusion path to files recognized by this lens;
- `excl` nodes for each path to be excluded from this lens.

```
augtool> print /augeas/load/Pam/
/augeas/load/Pam
/augeas/load/Pam/lens = "@Pam"
/augeas/load/Pam/incl = "/etc/pam.d/*"
/augeas/load/Pam/excl[1] = "*.augnew"
/augeas/load/Pam/excl[2] = "*.augsave"
/augeas/load/Pam/excl[3] = "*.dpkg-dist"
/augeas/load/Pam/excl[4] = "*.dpkg-bak"
/augeas/load/Pam/excl[5] = "*.dpkg-new"
/augeas/load/Pam/excl[6] = "*.dpkg-old"
/augeas/load/Pam/excl[7] = "*.rpmsave"
/augeas/load/Pam/excl[8] = "*.rpmnew"
/augeas/load/Pam/excl[9] = "*~"
```

Listing 5.3: Listing metadata for the Pam module

This tree can be manipulated to fine tune the lenses known by Augeas for a session, as well as the files parsed in the session. When the `/augeas/load` tree is modified, you have to call `load` again for the changes to take effect.

Let us look at some use cases.

Using only one lens

It is common to use Augeas to modify only one file. In that case you know exactly which lens you want to use and on which file. For performance reasons, you might want to narrow the lenses and files Augeas knows about. For example, if you want to only modify `/etc/fstab`, using the `Fstab` lens. In order to do that, we can start `augtool` without loading any lenses:

```
$ augtool --noautoload
augtool> print /augeas/load
/augeas/load
```

Listing 5.4: The effect of –noautoload on /augeas/load

This can also be achieved using the `AUG_NO_MODL_AUTOLOAD` flag with the API

The `print` command shows us that no lenses are known in the session. We can now tell Augeas to load the `Fstab` lens and to include `/etc/fstab` for it:

```
augtool> set /augeas/load/Fstab/lens ”Fstab.lns”
augtool> set /augeas/load/Fstab/incl ”/etc/fstab”
augtool> print /augeas/load
/augeas/load
/augeas/load/Fstab
/augeas/load/Fstab/lens = ”Fstab.lns”
/augeas/load/Fstab/incl = ”/etc/fstab”
```

Listing 5.5: Setting the Fstab lens manually in /augeas/load

We can now call `load` and list the files in `/files/etc`:

```
augtool> load
augtool> ls /files/etc
fstab/ = (none)
```

Listing 5.6: Loading files manually

Lenses loaded automatically have a `lens` statement which begins with a `@`, such a `@Fstab`. When you set the lens manually however, you have to specify the lens to use, for example `Fstab.lns`. See *Writing Your Own Lenses* on page 33 for more information on writing lenses.

Parsing a specific file

Augeas lenses have hardcoded lists of files they know about. For example the `Fstab` lens has an include statement for `/etc/fstab` hardcoded in `fstab.aug`. While Augeas attempts to cover the most common needs for inclusions, it cannot know about all files you are using. Some lenses don't even have default include statements because no common files are known to use them. This is the case of the `Json` lens, which is useful but applies to no common configuration file.

So how do you go about using the `Json` lens on a JSON file? You can modify the `/augeas/load` tree for that. For example if you have a `foo.json` file in your current directory, you could do the following:

This technique can be combined with the above to load only the `Json` module

```
$ augtool --root .
```

```
augtool> set /augeas/load/Json/incl "/foo.json"
augtool> load
augtool> ls /files
foo.json/ = (none)
```

Listing 5.7: Using the Json lens with /augeas/load

The files tree

The `/augeas/files` provides metadata about the files parsed by Augeas. The paths in this tree mirror thoses of the `/files` tree.

For each file, the following nodes may be present.

The path node

`path` is the path to the file data in the `/files` tree.

The mtime node

`mtime` is the last modification time of the file when it was read. Augeas uses this information internaly to speed up loading of files. Only the files whose modification time has changed are read again when `aug_load` is called.

The lens tree

The `lens` tree indicates the lens used to parse this file, as specified in the `/augeas/load` tree (see above). The `lens/info` node gives the path to the lens module (physically), as well as the position of the lens declaration in the file.

The error tree

When Augeas fails to parse a file, the parsing error is listed here.

This tree contains several nodes:

- `pos` is the position in the file, relative to the beginning, where Augeas failed to parse;
- `line` is the line in the file where Augeas failed to parse;
- `char` is the character of the line where Augeas failed to parse;
- `lens` is the lens that failed to parse. It is usually the same as as `lens/info` node listed above;
- `message` is the error message yielded by Augeas.

See *Troubleshooting Augeas* on page 41 for more information on interpreting the error messages

Example

In the example above, we see the that `/etc/ldap.conf` uses the `@Spacevars` lens, located in `spacevars.aug` on line 37, between characters 23 et 46.

The parsing of `/etc/ldap.conf` failed on position 9510, which located in beginning of line 310. The error message indicates that the file could not be fully parsed.

The variables tree

When you set variables in Augeas[1] the paths of the variables are recorded here.

[1]See *Using variables in paths* on page 16.

```
augtool> print /augeas/files/etc/ldap.conf/
/augeas/files/etc/ldap.conf
/augeas/files/etc/ldap.conf/path = "/files/etc/ldap.conf"
/augeas/files/etc/ldap.conf/mtime = "1298365882"
/augeas/files/etc/ldap.conf/lens = "@Spacevars"
/augeas/files/etc/ldap.conf/lens/info = \
   "/usr/share/augeas/lenses/dist/spacevars.aug:37.23-.46:"
/augeas/files/etc/ldap.conf/error = "parse_failed"
/augeas/files/etc/ldap.conf/error/pos = "9510"
/augeas/files/etc/ldap.conf/error/line = "310"
/augeas/files/etc/ldap.conf/error/char = "0"
/augeas/files/etc/ldap.conf/error/lens = \
   "/usr/share/augeas/lenses/dist/spacevars.aug:37.23-.46:"
/augeas/files/etc/ldap.conf/error/message = \
   "Iterated lens matched less than it should"
```

Listing 5.8: Inspecting ldap.conf metadata

Example:

```
augtool> defvar l /augeas/files/etc/ldap.conf/
augtool> print /augeas/variables/
/augeas/variables
/augeas/variables/l = "/augeas/files/etc/ldap.conf"
```

Listing 5.9: Defined variables are listed in /augeas/variables

As of Augeas 0.8.0, this node is purely informative: changing its value has no effect on the way Augeas works.

The span node

The /augeas/span node indicates whether the span functionality[2] is activated in the session.

[2]See *Locating nodes in files* on page 7.

6 Using Augeas in Puppet

Let it be observed, that slovenliness is no part of religion;
that neither this nor any text of Scripture,
condemns neatness of apparel.
Certainly this is a duty, not a sin,
“Cleanliness is, indeed, next to godliness.”

Sermon xciii, ‘on Dress’
John Wesley

Because Augeas is a configuration API, it fits right into tools that are made for configuration management. One of the most widely used of these tools in the open-source world is Puppet, and Augeas has been available as a native type in Puppet since version 0.24.7.

Since Puppet is written in Ruby, the Augeas Puppet type makes use of the Ruby bindings for Augeas.

The Augeas type

Puppet provides a native Augeas type since version 0.24.7.

The Augeas type in Puppet takes a list of commands labeled “changes”. The example of listing 1.6[1] then becomes:

```
augeas { ”hosts_alice”:
   changes => [
      ”set /files/etc/hosts/1/canonical alice”,
   ],
}
```

The `changes` attribute is an array of Augeas commands, similar to what you would pass to `augtool`.

It is recommended to use `augtool` to prepare and test the commands before you use them in Puppet.

Each call to the Augeas type starts a new Augeas session. The `save` call is ran automatically at the end of each session.

Setting a context

Proper quoting

While quoting in `augtool` is strict, quoting in Puppet can be tricky.

[1]See *Dropping into an interactive session* on page 10.

Puppet and idempotence

Idempotence is very important in configuration management tools such as Puppet. The Augeas type provides a `onlyif` statement to make it easy to ensure that Augeas is only called when necessary.

```
augeas { ”hosts_alice”:
   context => ”/files/etc/hosts/1”,
   changes => [
      ”set canonical alice”,
   ],
   onlyif => ”match canonical[. = ’alice’] size == 0”,
}
```

For proper idempotence, this statement has to be coupled with the methods described earlier[a].

[a]See *Ensuring idempotence* on page 16.

7 Writing Your Own Lenses

Augeas comes with a set of various lenses which cover most of the basic configuration files on a Unix machine. However, there are so many configuration file formats on Unix systems, that you are very likely to miss one at some point.

Augeas lenses are written in a ML language that is similar to OCaml. The language consists mostly of regexps and operators to combine them.

A simple example

Since Augeas lenses are mostly a combination of regular expressions that are often complex and fragile, it is safer to consider writing unit tests for each lens to ensure non-regression and confirm that all known cases are met by the lens. Our work will thus begin with the writing of a unit test, which will specify the way we will map the configuration entries to the Augeas tree. For extended information on unit tests, see the end of this chapter.

Unit Test

Example of a simple key/value conffilem, step by step

Module

Example of a simple key/value conffile, step by step

Regular expressions

The bidirectional nature of the Augeas language imposes strict conditions on the language[1]. This makes complex regular expressions languages such as PCRE hard to implement. For this reason, Augeas only supports POSIX simple regular expressions.

Give Examples

Special keywords

The Augeas language provides a set of keywords to build lenses.

[1]See chapter 2 on page **??**

key

label

store

value

seq

rec

square

Combination Operators

Augeas lenses are put together by assembling regular expressions with combination operators.

Concatenation Operator

Union Operator

Filters and Autoload

Augeas lenses need to specify which files they apply to. If they didn't, Augeas would have no way to know which lens to apply to which files. Trying to guess would be a really bad idea. For example, consider a file whose only content is the following:

```
# this is a comment
```

Many lenses are able to parse this line, and will mostly likely map it the same way. However, once a lens has been chosen for the file, the rest of the configuration statements are likely to be very different from one lens to another, so you are almost sure that the lens you chose will be wrong.

Each lenses may have one and only one autoload statement, involving a lens and a filter, such as the following:

```
autoload xfm
let lns = ...
let filter = incl ”/etc/foo.conf”
let xfm = transform lns filter
```

Typechecking lenses

Augeas comes with a command line tool called `augparse` which can be used to typecheck lenses, checking that they meet the conditions to be used as bidirectional transforms.

Typechecking recursive lenses

Unit tests

We have mentionned the importance of unit tests in the beginning of this chapter. It is worth repeting it: unit tests are essential to the stability of an Augeas lens. Unit tests need to be well written and kept up-to-date with new features and bug fixes to ensure that the lens continues to work with the files it was written for.

Augeas provides keywords to achieve unit tests in both the get and put directions.

Using Generic Modules

Augeas provides special modules to ease the writing of lenses.

The Util module

The Util (`util.aug`) module provides definitions of comments, empty lines and other utilities.

List functions and give examples.

The Sep module

The Sep (`sep.aug`) module provides definitions for separators. **List functions** and give examples.

> `Sep.opt_space` is a synonym for `Util.indent`. Both are strictly equivalent, but it is clearer to use the former as a separator and the latter as an indentation.

The Rx module

The Rx (`rx.aug`) module provides definitions for usual regular expressions. **List functions** and give examples.

The Build module

The Build (`build.aug`) module provides definitions for usual constructions of regular expression. **List functions** and give examples.

The IniFile module

INI files are quite standard even on Unix systems. However, there are many different implementations and variations. The Inifile (`inifile.aug`) module provides definitions to ease the writing of lenses for specific INI files. It is used as a basis for lenses such as Php (`php.aug`), MySQL (`mysql.aug`) or Puppet (`puppet.aug`). **List functions** and give examples.

Using your lens

Augeas uses a search path to find its lenses. By default, it will search for lenses in `$prefix/share/augeas/lenses` and `/$prefix/share/augeas/lenses/dist`, where `$prefix` is the compilation prefix, usually `/usr`.

The `dist` subdirectory is reserved for stock lenses, while the top directory can be used to store your own lenses.

If you prefer to store your lenses in another place, or just wish to try a new lens without installing it in your system, you can override this search path in several ways.

Ignoring the stock modules

In order to ignore the default search path for lenses, you can use the `--nostdinc` flag in `augtool`.

Adding your own directory of lenses

Directories containing additional lenses can be added to the search path by using the `--include` option in `augtool`, or the `AUGEAS_LENS_LIB` environment variable:

```
$ augtool --include mylenses
```

Documenting your modules

Talk about using NaturalDocs to document modules

Lens optimization

Augeas lenses are compiled into regular expressions. Some of these regular expressions can become very complex and typechecking them can exhaust both your CPU and memory.

These are simple rules to optimize your lenses.

Use standard constructions

Many configuration files have similar syntaxes. It is recommended to use the standard libraries to build new lenses.

Avoid regexp ”substractions”

Substractions of regexps are very costly, since they generate very complex regexps.

Group keys in blocks

The union of two similar blocks is usually more costly than a single merged block.

For example, the following:

```
let entry = Build.key_value kw1 Sep.equal (store Rx.word)
          | Build.key_value kw2 Sep.equal (store Rx.word)
```

will certainly be more efficient as:

```
let entry = Build.key_value (kw1 | kw2) Sep.equal (store Rx.word)
```

8 Troubleshooting Augeas

The Augeas tree is built using bidirectional grammars called lenses[1]. The configuration files will not appear in the Augeas tree if the lens responsible for parsing them fails to do so.

In the other direction[2], lenses may fail to save a tree back to a configuration file if that tree doesn't fit in the given lens.

Whatever you are trying to troubleshoot, you will most likely benefit from the metadata exposed in the `/augeas` node at the top of the Augeas tree.

A simple way to list all known errors in an augtool session is to type:

```
augtool> print /augeas//error
```

The double slash tells Augeas to search for all subnodes under `/augeas` whose label matches "error". The print command will return all subnodes of the matching nodes, given you the details of the errors.

If you want to see the error on a specific file, you can use the path

[1]See *Bidirectional transformations* on page 11.
[2]The `put` direction; See *Bidirectional transformations* on page 11

to that file in the expression. For example, to see the error on `/etc/fstab`, you can use:

```
augtool> print /augeas/files/etc/fstab/error
```

Files don't appear in the tree

There can be several reasons for a file to not appear in the Augeas tree.

No lens for the file

One possibility is that there is no existing lens for this file, or the lens you expect to parse this file has no filter for this file at this location. See chapter 7 on page 33 for more information on writing lenses.

UID has no rights to read

Another possibility is that the Unix UID you are using has no right to see the file. The "error" node in the `/augeas` tree will tell you so, with a message such as:

```
/augeas/files/etc/sudoers/error = "read_failed"
/augeas/files/etc/sudoers/error/message = "Permission denied"
```

Parsing failed

The last possibility is that the lens failed to parse part of the file, or the whole file.

Parsing errors are quite common, and there can be several reasons for them:

- The file uses `\r` for newlines. Most lenses, having been made for Unix systems, only recognize `\n` as valid newlines. Getting the file through dos2unix and trying again can confirm this possibility.
- The lens fails to parse a part of the file, for example it doesn't cover a specific case that is valid for this configuration file.
- The lens fails to parse the entire file.

In the last two cases, it is important to check that the configuration file is indeed valid. When available, use a command line tool provided with the application owning the configuration file, such as apachectl or visudo:

```
$ apachectl configtest
```

```
$ visudo -c
```

Note that when the application owning the configuration file is happy with the file and Augeas is not, it is always safer to consider that Augeas is wrong and that the lens has to be modified, since other users are likely to be in the same situation.

Save failed

Just as files can fail to be parsed by Augeas, trees can fail to be transformed back into files, too. This prevents Augeas from saving a tree that wouldn't make sense in the configuration file, thus preventing it from breaking configuration files.

Explain cases and solutions

Turning on debug

Augeas has a debug facility that is turned off by default. Two environment variables control the activate of this functionality: `AUGEAS_DEBUG` and `AUGEAS_DEBUG_DIR`.

9 Contacting the Augeas team

Tell me and I forget.
Teach me and I remember.
Involve me and I learn.

Benjamin Franklin

Augeas is an open-source project with an active community of users and developers.

There are several ways to contact the Augeas team:

- The augeas-devel mailing list at https://www.redhat.com/mailman/listinfo/augeas-devel;
- The IRC channel `#augeas` on the Freenode IRC network.

Contributing

You are very welcome to contribute code to the Augeas project.

Forking the git repository

Coding style

Sending patches

Getting support

Reporting bugs

GNU Free Documentation Licens

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or

XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies

3. COPYING IN QUANTITY

general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

ADDENDUM: How to use this License for your documents

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

List of listings