Summain—deterministic file manifests

Lars Wirzenius

2021-09-26 08:16

Contents

1 Introduction

A file manifest lists files, with their metadata.

To verify a backup has been restored correctly, one can compare a manifest of the data before the backup and after it has been restored. If the manifests are identical, the data has been restored correctly.

This requires a way to produce manifests that is deterministic: if run twice on the same input files, without the files having changed, the result should be identical. The Summain program does this.

This version of Summain has been written in Rust for the Obnam[1] project.

[1] `https://obnam.org/`

1.1 Why not mtree?

mtree[2] is a tool included in NetBSD[3] Unix since version 1.2, released in 1996. It produces a manifest, and can check a manifest against the file system. It is, in principle, a tool that solves the same problem Summain. Why not use an existing tool. Some reasons:

- I'm an anti-social not-invented-here jerk.
- It's an old C program, without tests in the source tree.
- The file format is custom, and not nice for reading by humans.
- It doesn't handle Unicode well.
  - a filename of `ö` is encoded as `\M-C\M-6`
  - but at least it can handle non-ASCII characters!
- It doesn't handle file metadata that's Linux specific.
  - extended attributes
  - the ext4 immutable bit
- It's single-threaded.

In principle, there is no reason why mtree couldn't be extended to support everything I need for Obnam. In practice, since I'm working on this in my free time in order to have fun, I prefer to write a new tool in Rust.

1.2 Why not use the old Python version of Summain

I don't like Python anymore. The old tool would need updates to work with current Python, and I'd rather use Rust.

2 Usage

Summain is given one or more files or directories on the command line, and it outputs to its standard output a manifest. If the command line arguments are the same, and the files haven't changed, the manifest is the same.

The output is YAML. Each file gets its own YAML document, delimieted by `---` and `...` as usual.

Summain does not itself traverse directories. Instead, a tool like **find**(1) should be used. Summain will, however, sort its command line arguments so that it doesn't matter if they're always in the same order.

3 Acceptance criteria

These scenarios verify that Summain handles the various kinds of file system objects it may encounter, with two exceptions: block and character devices. To create those, one needs to be the `root` user, and we don't want to have to run

[2] `http://cdn.netbsd.org/pub/pkgsrc/current/pkgsrc/pkgtools/mtree/README.html`
[3] `https://en.wikipedia.org/wiki/NetBSD`

the test suite as root. Instead, we blithely rely on the output being correct for those anyway. Testing manually indicates that it works, and the only difference from, say, regular files is that the mode starts with a `b` or `c`, which is exactly correct.

3.1 Directory

given an installed summain
given directory **empty**
and mtime for **empty** is **456**
when I run **chmod a=rx empty**
when I run **summain empty**
then output matches file **empty.yaml**

File: **empty.yaml**

```
---
path: empty
mode: dr-xr-xr-x
mtime: 456
mtime_nsec: 0
nlink: 2
size: ~
sha256: ~
target: ~
```

3.2 Writeable file

given an installed summain
given file **foo**
and mtime for **foo** is **22**
when I run **chmod a=rw foo**
when I run **summain foo**
then output matches file **foo.yaml**

File: **foo.yaml**

```
---
path: foo
mode: "-rw-rw-rw-"
mtime: 22
mtime_nsec: 0
nlink: 1
size: 0
sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
target: ~
```

3.3 Read-only file

given an installed summain
given file **foo**
and mtime for **foo** is **44**
when I run **chmod a=r foo**
when I run **summain foo**
then output matches file **readonly.yaml**

File: **readonly.yaml**

```
---
path: foo
mode: "-r--r--r--"
mtime: 44
mtime_nsec: 0
nlink: 1
size: 0
sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
target: ~
```

3.4 Two files sorted

given an installed summain
given file **aaa**
and mtime for **aaa** is **44**
given file **bbb**
and mtime for **bbb** is **44**
when I run **chmod a=r aaa bbb**
when I run **summain bbb aaa**
then output matches file **aaabbb.yaml**

File: **aaabbb.yaml**

```
---
path: aaa
mode: "-r--r--r--"
mtime: 44
mtime_nsec: 0
nlink: 1
size: 0
sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
target: ~
---
path: bbb
mode: "-r--r--r--"
mtime: 44
mtime_nsec: 0
```

```
nlink: 1
size: 0
sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
target: ~
```

3.5 Symlink

given an installed summain
given symlink **ccc** pointing at **aaa**
and mtime for **ccc** is **44**
when I run **summain ccc**
then output matches file **ccc.yaml**

File: **ccc.yaml**

```
---
path: ccc
mode: lrwxrwxrwx
mtime: 44
mtime_nsec: 0
nlink: 1
size: 3
sha256: ~
target: aaa
```

3.6 Unix domain socket

given an installed summain
given socket **aaa**
and file **aaa** has mode **0700**
and mtime for **aaa** is **44**
when I run **summain aaa**
then output matches file **socket.yaml**

File: **socket.yaml**

```
---
path: aaa
mode: srwx------
mtime: 44
mtime_nsec: 0
nlink: 1
size: 0
sha256: ~
target: ~
```

3.7 Named pipe

given an installed summain
given named pipe **aaa**
and file **aaa** has mode **0700**
and mtime for **aaa** is **44**
when I run **summain aaa**
then output matches file **fifo.yaml**

File: **fifo.yaml**

```
---
path: aaa
mode: prwx------
mtime: 44
mtime_nsec: 0
nlink: 1
size: 0
sha256: ~
target: ~
```