recover-corrupted-blob-object

Date: Fri, 9 Nov 2007 08:28:38 -0800 (PST) From: Linus Torvalds <torvalds@linux-foundation.org> Subject: corrupt object on git-gc Abstract: Some tricks to reconstruct blob objects in order to fix a corrupted repository. Content-type: text/asciidoc

How to recover a corrupted blob object
==========

```
On Fri, 9 Nov 2007, Yossi Leybovich wrote:
>
> Did not help still the repository look for this object?
> Any one know how can I track this object and understand which file is it
```

So exactly **because** the SHA-1 hash is cryptographically secure, the hash itself doesn’t actually tell you anything, in order to fix a corrupt object you basically have to find the "original source" for it.

The easiest way to do that is almost always to have backups, and find the same object somewhere else. Backups really are a good idea, and Git makes it pretty easy (if nothing else, just clone the repository somewhere else, and make sure that you do **not** use a hard-linked clone, and preferably not the same disk/machine).

But since you don’t seem to have backups right now, the good news is that especially with a single blob being corrupt, these things **are** somewhat debuggable.

First off, move the corrupt object away, and **save** it. The most common cause of corruption so far has been memory corruption, but even so, there are people who would be interested in seeing the corruption - but it’s basically impossible to judge the corruption until we can also see the original object, so right now the corrupt object is useless, but it’s very interesting for the future, in the hope that you can re-create a non-corrupt version.

```
So:


> ib]$ mv .git/objects/4b/9458b3786228369c63936db65827de3cc06200 ../
```

This is the right thing to do, although it’s usually best to save it under it’s full SHA-1 name (you just dropped the "4b" from the result ;).

Let’s see what that tells us:

```
> ib]$ git-fsck --full
> broken link from    tree 2d9263c6d23595e7cb2a21e5ebbb53655278dff8
>              to    blob 4b9458b3786228369c63936db65827de3cc06200
> missing blob 4b9458b3786228369c63936db65827de3cc06200
```

Ok, I removed the "dangling commit" messages, because they are just messages about the fact that you probably have rebased etc, so they’re not at all interesting. But what remains is still very useful. In particular, we now know which tree points to it!

Now you can do

```
git ls-tree 2d9263c6d23595e7cb2a21e5ebbb53655278dff8
```

which will show something like

```
100644 blob 8d14531846b95bfa3564b58ccfb7913a034323b8    .gitignore
100644 blob ebf9bf84da0aab5ed944264a5db2a65fe3a3e883    .mailmap
100644 blob ca442d313d86dc67e0a2e5d584b465bd382cbf5c    COPYING
100644 blob ee909f2cc49e54f0799a4739d24c4cb9151ae453    CREDITS
040000 tree 0f5f709c17ad89e72bdbbef6ea221c69807009f6    Documentation
100644 blob 1570d248ad9237e4fa6e4d079336b9da62d9ba32    Kbuild
100644 blob 1c7c229a092665b11cd46a25dbd40feeb31661d9    MAINTAINERS
...
```

and you should now have a line that looks like

```
10064 blob 4b9458b3786228369c63936db65827de3cc06200 my-magic-file
```

in the output. This already tells you a **lot** it tells you what file the corrupt blob came from!

Now, it doesn’t tell you quite enough, though: it doesn’t tell what **version** of the file didn’t get correctly written! You might be really lucky, and it may be the version that you already have checked out in your working tree, in which case fixing this problem is really simple, just do

```
git hash-object -w my-magic-file
```

again, and if it outputs the missing SHA-1 (4b945..) you’re now all done!

But that’s the really lucky case, so let’s assume that it was some older version that was broken. How do you tell which version it was?

The easiest way to do it is to do

```
git log --raw --all --full-history -- subdirectory/my-magic-file
```

and that will show you the whole log for that file (please realize that the tree you had may not be the top-level tree, so you need to figure out which subdirectory it was in on your own), and because you're asking for raw output, you'll now get something like

```
commit abc
Author:
Date:
  ..
:100644 100644 4b9458b... newsha... M  somedirectory/my-magic-file

commit xyz
Author:
Date:

  ..
:100644 100644 oldsha... 4b9458b... M   somedirectory/my-magic-file
```

and this actually tells you what the **previous** and **subsequent** versions of that file were! So now you can look at those ("oldsha" and "newsha" respectively), and hopefully you have done commits often, and can re-create the missing my-magic-file version by looking at those older and newer versions!

If you can do that, you can now recreate the missing object with

```
git hash-object -w <recreated-file>
```

and your repository is good again!

(Btw, you could have ignored the fsck, and started with doing a

```
git log --raw --all
```

and just looked for the sha of the missing object (4b9458b..) in that whole thing. It's up to you - Git does **have** a lot of information, it is just missing one particular blob version.

Trying to recreate trees and especially commits is **much** harder. So you were lucky that it's a blob. It's quite possible that you can recreate the thing.

```
Linus
```

□□□ #2
□ □□□ □□□ 2026-10-31 20:14:59 UTC
□ □□□ □□□ 2026-10-31 20:15:00 UTC