Ruby - Bug #9790

Zlib::GzipReader only decompressed the first of concatenated files

04/30/2014 08:56 PM - quainjn (Jake Quain)

Status:	Closed		
Priority:	Normal		
Assignee:	drbrain (Eric Hodel)		
Target version:			
ruby -v:	2.1.1	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

Description

There is a similar old issue in Node that I came across that perfectly describes the situation in ruby:

https://github.com/joyent/node/issues/6032

In ruby given the following setup:

```
echo "1" > 1.txt
echo "2" > 2.txt
gzip 1.txt
gzip 2.txt
cat 1.txt.gz 2.txt.gz > 3.txt.gz
```

Calling:

```
Zlib::GzipReader.open("3.txt.gz") do |gz|
  print gz.read
end
```

would just print:

```
1
```

Related issues:

Related to Ruby - Bug #14804: GzipReader cannot read Freebase dump (but gzcat...	**Closed**
Has duplicate Ruby - Bug #11180: Missing lines with Zlib::GzipReader	**Closed**

Associated revisions

Revision d52dffd817d9285f7600138e2f69f46891fff845 - 09/14/2020 08:52 AM - jeremyevans (Jeremy Evans)

[ruby/zlib] Add Zlib::GzipReader.zcat for handling multiple gzip streams in gz file

Most gzip tools support concatenated gz streams in a gz file. This
offers a way to handle such gz files in Ruby.

Fixes [Bug #9790]
Fixes [Bug #11180]
Fixes [Bug #14804]

https://github.com/ruby/zlib/commit/e2ce56de7d

Revision d52dffd817d9285f7600138e2f69f46891fff845 - 09/14/2020 08:52 AM - jeremyevans (Jeremy Evans)

[ruby/zlib] Add Zlib::GzipReader.zcat for handling multiple gzip streams in gz file

Most gzip tools support concatenated gz streams in a gz file. This
offers a way to handle such gz files in Ruby.

Fixes [Bug #9790]
Fixes [Bug #11180]
Fixes [Bug #14804]

https://github.com/ruby/zlib/commit/e2ce56de7d

Revision d52dffd8 - 09/14/2020 08:52 AM - jeremyevans (Jeremy Evans)

[ruby/zlib] Add Zlib::GzipReader.zcat for handling multiple gzip streams in gz file

Most gzip tools support concatenated gz streams in a gz file. This
offers a way to handle such gz files in Ruby.

Fixes [Bug #9790]
Fixes [Bug #11180]
Fixes [Bug #14804]

https://github.com/ruby/zlib/commit/e2ce56de7d

History

#1 - 05/01/2014 10:53 PM - drbrain (Eric Hodel)

- Category set to ext

- Status changed from Open to Assigned

- Assignee set to drbrain (Eric Hodel)

- Target version set to 2.2.0

#2 - 02/24/2015 04:17 PM - akostadinov (Aleksandar Kostadinov)

Because gzip format allows multiple entries with filename I'd suggest to support a method like Java's ZipInputStream getNextEntry() [1]. This way programmer can choose to read everything as one chunk of data or multiple chunks each with its own name. This would allow storing and then retrieving multiple files in/from one gz.

On the other hand the command line gzip utility only supports reading the whole thing as one. So a convenience method to read everything in one go, would also be nice.

[1] http://docs.oracle.com/javase/7/docs/api/java/util/zip/ZipInputStream.html

#3 - 02/25/2015 06:16 AM - duerst (Martin Dürst)

Aleksandar Kostadinov wrote:

> Because gzip format allows multiple entries with filename I'd suggest to support a method like Java's ZipInputStream getNextEntry() [1]. This way programmer can choose to read everything as one chunk of data or multiple chunks each with its own name. This would allow storing and then retrieving multiple files in/from one gz.

Good idea, but it should be more Ruby-like, such as .each_file or so.

#4 - 05/26/2015 09:37 AM - exAspArk (Evgeny Li)

Hey guys, is there any updates?

I have created a small gem yesterday to make it able to read multiple files https://github.com/exAspArk/multiple_files_gzip_reader

```
> MultipleFilesGzipReader.open("3.txt.gz") do |gz|
>   puts gz.read
> end

# 1
# 2
# => nil
```

#5 - 05/27/2015 12:45 AM - nagachika (Tomoyuki Chikanaga)

- Has duplicate Bug #11180: Missing lines with Zlib::GzipReader added

#6 - 10/29/2019 03:35 AM - jeremyevans0 (Jeremy Evans)

- Related to Bug #14804: GzipReader cannot read Freebase dump (but gzcat/zless can) added

#7 - 11/27/2019 03:38 PM - jeremyevans0 (Jeremy Evans)

- File zlib-gzreader-each_file-9790.patch added

Attached is a patch that adds Zlib::GzipReader.each_file will which handle multiple concatenated gzip streams in the same file, similar to common tools that operate on .gz files. Zlib::GzipReader.each_file yields one Zlib::GzipReader instance per gzip stream in the file.

#8 - 01/14/2020 04:10 AM - ko1 (Koichi Sasada)

- Status changed from Assigned to Feedback

do you have real (popular) usecases?

#9 - 01/14/2020 04:33 AM - jeremyevans0 (Jeremy Evans)

ko1 (Koichi Sasada) wrote:

> do you have real (popular) usecases?

For real but not necessarily popular, but at least OpenBSD's package format uses this. I'm not sure if package formats for other operating systems use it, though. #14804 pointed out it was used by this file: http://commondatastorage.googleapis.com/freebase-public/rdf/freebase-rdf-latest.gz

There are 3 bug reports for this, and I think while the need isn't common, it's something worth supporting via a new method. However, if we decide we don't want to support this, I'm fine closing the 3 bug reports.

#10 - 01/16/2020 07:39 AM - ko1 (Koichi Sasada)

- mame: can each_file return an Enumerator? Seems difficult to implement it
- matz: How about always behaving like zcat? Is an option to keep the old behaviors really needed?
- akr: The traditional behavior should be kept
- akr: gzip(1) describes concatenation of gzip files in ADVANCED USAGE. https://www.gnu.org/software/gzip/manual/gzip.html#Advanced-usage

Conclusion:

- matz: it should behave like zcat. Handling each member should be deleted.

#11 - 01/16/2020 02:32 PM - Dan0042 (Daniel DeLorme)

> matz: it should behave like zcat. Handling each member should be deleted.

Really? I agree that Zlib::GzipReader.open should behave like zcat, but each_file is a useful additional feature to have. And it's already implemented.

#12 - 01/17/2020 04:51 AM - shyouhei (Shyouhei Urabe)

Dan0042 (Daniel DeLorme) wrote:

> > matz: it should behave like zcat. Handling each member should be deleted.
>
> Really? I agree that Zlib::GzipReader.open should behave like zcat, but each_file is a useful additional feature to have. And it's already implemented.

I was at the meeting @ko1 (Koichi Sasada) is talking about. Devs at the meeting were not sure if "each_file is a useful additional feature to have" you say is real or not.

Do you know if there are cases when taking a random member of such gzip file is actually useful?

#13 - 01/17/2020 02:17 PM - Dan0042 (Daniel DeLorme)

I can't think of taking a random member but I can imagine wanting to extract to separate files.

```
Zlib::GzipReader.each_file("input.gz").with_index do |gzip,i|
  File.open("output#{i}","w"){ |f| f.write(gzip.read) }
end
```

But I admit I haven't needed this personally, so maybe this is not so useful.

#14 - 01/17/2020 04:04 PM - jeremyevans0 (Jeremy Evans)

shyouhei (Shyouhei Urabe) wrote:

> Do you know if there are cases when taking a random member of such gzip file is actually useful?

It's not taking a random member, it's processing the separate gzip streams in order, which is actually useful. For example, in OpenBSD's packages, the first gzip stream in the package contains the package metadata. That is read completely, and after processing the metadata, the programs that handle packages determine whether it needs to read the actual package data (stored in the second gzip stream). If it doesn't need to process the package data, it can then stop at that point without having read any more than it needs to.

#15 - 01/18/2020 12:51 AM - nobu (Nobuyoshi Nakada)

jeremyevans0 (Jeremy Evans) wrote:

> For example, in OpenBSD's packages, the first gzip stream in the package contains the package metadata. That is read completely, and after processing the metadata, the programs that handle packages determine whether it needs to read the actual package data (stored in the second gzip stream). If it doesn't need to process the package data, it can then stop at that point without having read any more than it needs to.

It is an interesting usage.
Does that metadata need to be a separate member, not the leading part of a large stream?

#16 - 01/18/2020 01:07 AM - jeremyevans0 (Jeremy Evans)

nobu (Nobuyoshi Nakada) wrote:

> It is an interesting usage.
> Does that metadata need to be a separate member, not the leading part of a large stream?

It's easier and more efficient if the metadata is a separate member. I'm guessing it could be changed to use a single large stream, but there is no reason to do so, and doing so would break existing tooling.

#17 - 06/02/2020 04:17 PM - jeremyevans0 (Jeremy Evans)

ko1 (Koichi Sasada) wrote in #note-10:

> - mame: can each_file return an Enumerator? Seems difficult to implement it
> - matz: How about always behaving like zcat? Is an option to keep the old behaviors really needed?
> - akr: The traditional behavior should be kept
> - akr: gzip(1) describes concatenation of gzip files in ADVANCED USAGE.
>   https://www.gnu.org/software/gzip/manual/gzip.html#Advanced-usage
>
> Conclusion:
>
> - matz: it should behave like zcat. Handling each member should be deleted.

I'm not sure, but it seems the proposal here would be to make all Zlib::GzipReader methods transparently handle multiple streams. There are two issues with doing that:

1. It's very invasive, all methods would need to change, some in fairly complex ways.

2. More importantly, it would break cases where non-gzip data was stored after gzip data. Currently you can use GzipReader in such cases, and the io pointer after the gzip processing will be directly after where the gzip stream ends, at which point you can use the io normally. Basically, if you make this change, you could no longer embed a gzip stream in the middle of a file and read just that stream.

If we don't want to add Zlib::GzipReader.each_file but we want to add something like zcat, here's a pull request that implements Zlib::GzipReader.zcat: https://github.com/ruby/zlib/pull/13. I think Zlib::GzipReader.each_file is a more useful and flexible method than Zlib::GzipReader.zcat. We could certainly have both, though.

#18 - 07/22/2020 03:39 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Feedback to Closed

matz approved Zlib::GzipReader.zcat, so I merged the pull request into zlib.

Files

zlib-gzreader-each_file-9790.patch	3.47 KB	11/27/2019	jeremyevans0 (Jeremy Evans)