

Re: [RFC PATCH 0/5] Union Mount: A Directory listing approach with lseek support

Re: [RFC PATCH 0/5] Union Mount: A Directory listing approach with lseek support

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-12/msg01882.html>

- *From:* Bharata B Rao <bharata@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 6 Dec 2007 20:40:12 +0530
-

On Thu, Dec 06, 2007 at 11:01:18AM +0100, Jan Blunck wrote:

On Wed, Dec 05, Dave Hansen wrote:

I think the key here is what kind of consistency we're trying to provide. If a directory is being changed underneath a reader, what kinds of guarantees do they get about the contents of their directory read? When do those guarantees start? Are there any at open() time?

But we still want to be compliant to what POSIX defines. The problem isn't the consistency of the readdir result but the seekdir/telldir interface. IMHO that interface is totally broken: you need to be able to find every offset given by telldir since the last open. The problem is that seekdir isn't able to return errors. Otherwise you could just forbid seeking on union directories.

Also, what kind of consistency is expected when a directory is open(2)ed and readdir(2) and lseek(2) are applied to it when the directory gets changed underneath the reader. From this: <http://www.opengroup.org/onlinepubs/009695399/functions/lseek.html> the behaviour/guarantees wasn't apparent to me.

Rather than give each `_dirent_` an offset, could we give each sub-mount an offset? Let's say we have three members comprising a union mount directory. The first has 100 dirents, the second 200, and the third 10,000. When the first readdir is done, we populate the table like this:

```
mount_offset[0] = 0;
mount_offset[1] = 100;
mount_offset[2] = 300;
```

If someone seeks back to 150, then we subtract the mount[1]'s offset

Re: [RFC PATCH 0/5] Union Mount: A Directory listing approach with lseek support

(100), and realize that we want the 50th dirent from mount[1].

Yes, that is a nice idea and it is exactly what I have implemented in my patch series. But you forgot one thing: directories are not flat files. The dentry offset in a directory is a random cookie. Therefore it is not possible to have a linear mapping without allocating memory.

And I defined this linear behaviour on the cache of dirents we maintain in the approach I posted. And the main reason we maintain cache of dirents in memory is for duplicate elimination.

I don't know whether we're bound to this:

<http://www.opengroup.org/onlinepubs/007908775/xsh/readdir.html>

"If a file is removed from or added to the directory after the most recent call to opendir() or rewinddir(), whether a subsequent call to readdir() returns an entry for that file is unspecified."

But that would seem to tell me that once you populate a table such as the one I've described and create it at open(dir) time, you don't actually ever need to update it.

Yes, I'm using such a patch on our S390 buildservers to work around some readdir/seek/rm problem with old glibc versions. It seems to work but on the other hand this are really huge systems and I haven't run out of memory while doing a readdir yet ;)

The proper way to implement this would be to cache the offsets on a per inode base. Otherwise the user could easily DoS this by opening a number of directories and never close them.

You mean cache the offsets or dirents ? How would that solve the seek problem ? How would it enable you to define a seek behaviour for the entire union of directories ?

Regards,
Bharata.

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Re: [RFC PATCH 0/5] Union Mount: A Directory listing approach with lseek support

Re: [RFC PATCH 0/5] Union Mount: A Directory listing approach with lseek support

Please read the FAQ at <http://www.tux.org/lkml/>