Subject: Re: [PATCH 01/10] Containers(V10): Basic container framework
Posted by Paul Menage on Wed, 30 May 2007 14:02:00 GMT
View Forum Message <> Reply to Message

On 5/30/07, Andrew Morton <akpm@linux-foundation.org> wrote:
>
> Holy cow, do we need all those?

I'll experiment to see which ones we can get rid of.

```
>
> > +typedef enum {
> > +	CONT_REMOVED,
> > +} container_flagbits_t;
>
> typedefs are verboten.  Fortunately this one is never referred to - only
> the values are used, so we can delete it.
```

OK.

>
> Taking dcache_lock in here is unfortunate. A filesystem really shouldn't
> be playing with that lock.

Is there a recommended way to do what I want to do, i.e. clear out all
the dentries from a virtual fs directory and rebuild them whilst
holding the directory's i_sem so no one can see the transiently empty
directory?

>
> The code's a bit short on comments.

I'll add some.

```
> > +	root = d_alloc_root(inode);
> > +	if (!root) {
> > +		iput(inode);
> > +		return -ENOMEM;
>
> I bet that iput() hasn't been tested ;)
```

Correct.

>
> People have hit unpleasant problems before now running iput() against
> partially-constructed inodes.

What kinds of problems? Are there bits of state that I should fully

construct even if I'm going to iput() it, or is there a better
function to call? fs/ext3/super.c seems to do the same thing.

```
> > +           if (ret)
> > +                   goto out_unlock;
>
> Did we just leak *root?
```

I believe we did. I'll fix that.

```
> >
> > +static inline void get_first_subsys(const struct container *cont,
> > +                               struct container_subsys_state **css,
> > +                               int *subsys_id) {
> > +    const struct containerfs_root *root = cont->root;
> > +    const struct container_subsys *test_ss;
> > +    BUG_ON(list_empty(&root->subsys_list));
> > +    test_ss = list_entry(root->subsys_list.next,
> > +                         struct container_subsys, sibling);
> > +    if (css) {
> > +            *css = cont->subsys[test_ss->subsys_id];
> > +            BUG_ON(!*css);
> > +    }
> > +    if (subsys_id)
> > +            *subsys_id = test_ss->subsys_id;
> > +}
>
> This ends up having several callers and its too large to inline.
```

Two large from a compiler PoV or from a style PoV? It's basically just
six dereferences and two comparisons, plus the BUG_ON()s.

```
>
> Do we actually want to support lseek on these things?
>
> If not we can leave this null and use nonseekable_open() in ->open.
```

I inherited that from cpusets without thinking about it too much. I
guess that we don't really need seekability.

```
> > +    } else if (S_ISREG(mode)) {
> > +            inode->i_size = 0;
> > +            inode->i_fop = &container_file_operations;
> > +    }
>
> The S_ISREG files have no ->i_ops?
```

Not currently. I don't see anything in inode_operations that we want to be able to do on non-directories.

Paul