Subject: Re: [RFC/PATCH 2/8]: CGroup Files: Add a cgroup write_string control file method
Posted by Paul Menage on Tue, 13 May 2008 21:01:01 GMT
View Forum Message <> Reply to Message

On Tue, May 13, 2008 at 1:07 PM, Andrew Morton
<akpm@linux-foundation.org> wrote:
> >
> > + /* If non-zero, defines the maximum length of string that can
> > + * be passed to write_string; defaults to 64 */
> > + int max_write_len;
>
> would size_t be a more appropriate type?
>

Probably overkill, but I guess it's technically more correct. Updated
for the next version of these patches.

>
> s/) (/)(/ would be more conventional.
>

OK, I've updated this and the other extraneous spaces in a separate patch.

>
> > + /*
> > + * write_string() is passed a nul-terminated kernelspace
> > + * buffer of maximum length determined by max_write_len
> > + */
> > + int (*write_string) (struct cgroup *cgrp, struct cftype *cft,
> > + char *buffer);
>
> Should these return size_t?

No, it returns 0 or a -ve error code. I've added a comment to this effect.
> > char *buffer = static_buffer;
> > - ssize_t max_bytes = sizeof(static_buffer) - 1;
> > + ssize_t max_bytes = cft->max_write_len ?: sizeof(static_buffer) - 1;
>
> A blank line between end-of-locals and start-of-code is conventional
> and, IMO, easier on the eye.
>
> Does gcc actually generate better code with that x?:y thing?

I doubt it - but I felt that it made the code a bit clearer since it
reduces repetition. I can change it to

size_t max_bytes = cft->max_write_len;

```
 if (!max_bytes)
  max_bytes = sizeof(static_buffer) - 1;
```

Paul