

Subject: Re: [PATCH 2/8] CGroup Files: Add write_string cgroup control file method
Posted by [akpm](#) on Tue, 24 Jun 2008 23:19:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Fri, 20 Jun 2008 16:44:00 -0700

menage@google.com wrote:

```
> This patch adds a write_string() method for cgroups control files. The
> semantics are that a buffer is copied from userspace to kernelspace
> and the handler function invoked on that buffer. The buffer is
> guaranteed to be nul-terminated, and no longer than max_write_len
> (defaulting to 64 bytes if unspecified). Later patches will convert
> existing raw file write handlers in control group subsystems to use
> this method.
>
> Signed-off-by: Paul Menage <menage@google.com>
>
> ---
> include/linux/cgroup.h | 14 +++++
> kernel/cgroup.c       | 35 +++++
> 2 files changed, 49 insertions(+)
>
> Index: cws-2.6.26-rc5-mm3/include/linux/cgroup.h
> =====
> --- cws-2.6.26-rc5-mm3.orig/include/linux/cgroup.h
> +++ cws-2.6.26-rc5-mm3/include/linux/cgroup.h
> @@ -205,6 +205,13 @@ struct cftype {
>  * subsystem, followed by a period */
>  char name[MAX_CFTYPE_NAME];
>  int private;
> +
> + /*
> + * If non-zero, defines the maximum length of string that can
> + * be passed to write_string; defaults to 64
> + */
> + size_t max_write_len;
> +
>  int (*open)(struct inode *inode, struct file *file);
>  ssize_t (*read)(struct cgroup *cgrp, struct cftype *cft,
>    struct file *file,
> @@ -249,6 +256,13 @@ struct cftype {
>  int (*write_s64)(struct cgroup *cgrp, struct cftype *cft, s64 val);
> +
> + /*
> + * write_string() is passed a nul-terminated kernelspace
> + * buffer of maximum length determined by max_write_len.
> + * Returns 0 or -ve error code.
> + */
```

```
> + int (*write_string)(struct cgroup *cgrp, struct cftype *cft,
> +     const char *buffer);
```

Everything seems to use size_t (or ssize_t?) except for the ->write_string return value. Can any of this be improved?

```
> + /*
>  * trigger() callback can be used to get some kick from the
>  * userspace, when the actual string written is not important
>  * at all. The private field can be used to determine the
> Index: cws-2.6.26-rc5-mm3/kernel/cgroup.c
> =====
> --- cws-2.6.26-rc5-mm3.orig/kernel/cgroup.c
> +++ cws-2.6.26-rc5-mm3/kernel/cgroup.c
> @@ -1363,6 +1363,39 @@ static ssize_t cgroup_write_X64(struct c
>  return retval;
>  }
>
> +static ssize_t cgroup_write_string(struct cgroup *cgrp, struct cftype *cft,
> +     struct file *file,
> +     const char __user *userbuf,
> +     size_t nbytes, loff_t *unused_ppos)
> +{
> + char local_buffer[64];
> + int retval = 0;
> + size_t max_bytes = cft->max_write_len;
> + char *buffer = local_buffer;
> +
> + if (!max_bytes)
> +     max_bytes = sizeof(local_buffer) - 1;
> + if (nbytes >= max_bytes)
> +     return -E2BIG;
> + /* Allocate a dynamic buffer if we need one */
> + if (nbytes >= sizeof(local_buffer)) {
> +     buffer = kmalloc(nbytes + 1, GFP_KERNEL);
> +     if (buffer == NULL)
> +         return -ENOMEM;
> + }
> + if (nbytes && copy_from_user(buffer, userbuf, nbytes))
> +     return -EFAULT;
> +
> + buffer[nbytes] = 0; /* nul-terminate */
> + strncpy(buffer, userbuf, nbytes);
> + retval = cft->write_string(cgrp, cft, buffer);
> + if (!retval)
> +     retval = nbytes;
> + if (buffer != local_buffer)
> +     kfree(buffer);
```

```
> + return retval;
> +}
> +
> static ssize_t cgroup_common_file_write(struct cgroup *cgrp,
>     struct cftype *cft,
>     struct file *file,
> @@ -1440,6 +1473,8 @@ static ssize_t cgroup_file_write(struct
>     return cft->write(cgrp, cft, file, buf, nbytes, ppos);
>     if (cft->write_u64 || cft->write_s64)
>     return cgroup_write_X64(cgrp, cft, file, buf, nbytes, ppos);
> + if (cft->write_string)
> + return cgroup_write_string(cgrp, cft, file, buf, nbytes, ppos);
>     if (cft->trigger) {
>         int ret = cft->trigger(cgrp, (unsigned int)cft->private);
>         return ret ? ret : nbytes;
>     }
> --
```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
