Subject: Re: [PATCH] kthread: Spontaneous exit support
Posted by Christoph Hellwig on Mon, 23 Apr 2007 11:25:37 GMT
View Forum Message <> Reply to Message

On Sun, Apr 22, 2007 at 09:12:55PM -0600, Eric W. Biederman wrote:
>
> This patch implements the kthread helper functions kthread_start
> and kthread_end which make it simple to support a kernel thread
> that may decided to exit on it's own before we request it to.
> It is still assumed that eventually we will get around to requesting
> that the kernel thread stop.

I don't think having to parallel APIs is a good idea, people will
get utterly confused which one to use. Better always grab a reference
in kthread_create and drop it in kthread_stop. For normal thread
no change in behaviour and only slightly more code in the slowpath.

Of course it will need an audit for half-assed kthread conversion
first to avoid task_struct reference count leaks.

In addition to that kthrad_end implementation look wrong. When
the kthread has exited prematurely no one will call complete
on kthread_stop_info.done before it's been setup. Interestingly
the comment there indicates someone thought about threads exiting
early, but it became defunkt during all the rewrites of the
kthread code.

```
> +/**
> + * kthread_start - create and wake a thread.
> + * @threadfn: the function to run until kthread_should_stop().
> + * @data: data ptr for @threadfn.
> + * @namefmt: printf-style name for the thread.
> + *
> + * Description: Convenient wrapper for kthread_create() followed by
> + * get_task_struct() and wake_up_process. kthread_start should be paired
> + * with kthread_end() so we don't leak task structs.
> + *
> + * Returns the kthread or ERR_PTR(-ENOMEM).
> + */
> +#define kthread_start(threadfn, data, namefmt, ...)      \
> +({            \
> + struct task_struct *__k         \
> +  = kthread_create(threadfn, data, namefmt, ## __VA_ARGS__); \
> + if (!IS_ERR(__k)) {         \
> +  get_task_struct(__k);        \
> +  wake_up_process(__k);        \
> + }          \
```

```
> + __k;          \
> +})
> +int kthread_end(struct task_struct *k);
>
>  static inline int __kthread_should_stop(struct task_struct *tsk)
>  {
> diff --git a/kernel/kthread.c b/kernel/kthread.c
> index 9b3c19f..d6d63c6 100644
> --- a/kernel/kthread.c
> +++ b/kernel/kthread.c
> @@ -179,6 +179,24 @@ int kthread_stop(struct task_struct *tsk)
>  }
>  EXPORT_SYMBOL(kthread_stop);
>
> +/**
> + * kthread_end - signal a kthread and wait for it to exit.
> + * @task: The kthread to end.
> + *
> + * Description: Convenient wrapper for kthread_stop() followed by
> + * put_task_struct().  Returns the kthread exit code.
> + *
> + * kthread_start()/kthread_end() can handle kthread that spontaneously exit
> + * before the kthread is requested to terminate.
> + */
> +int kthread_end(struct task_struct *task)
> +{
> + int ret;
> + ret = kthread_stop(task);
> + put_task_struct(task);
> + return ret;
> +}
> +EXPORT_SYMBOL(kthread_end);
```

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