

---

Subject: Re: [PATCH] mtd\_blkdevs: Convert to use the kthread API  
Posted by [David Woodhouse](#) on Sun, 22 Apr 2007 13:23:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Sun, 2007-04-22 at 13:24 +0100, Christoph Hellwig wrote:  
> This is the full conversion I sent to Dave in April 2006, but never  
> got any feedback to:

Sorry about that; I need prodding sometimes. I'll provide some now...

Can you show me why the thread won't now miss a wakeup if it goes to sleep just as a new request is added to its queue?

Having already applied Eric's patch, this is the delta to yours...

```
diff --git a/drivers/mtd/mtd_blkdevs.c b/drivers/mtd/mtd_blkdevs.c
index 1aa018a..d065dba 100644
--- a/drivers/mtd/mtd_blkdevs.c
+++ b/drivers/mtd/mtd_blkdevs.c
@@ -29,9 +29,7 @@ extern struct mutex mtd_table_mutex;
extern struct mtd_info *mtd_table[];

struct mtd_blkcore_priv {
- struct completion thread_dead;
- int exiting;
- wait_queue_head_t thread_wq;
+ struct task_struct *thread;
    struct request_queue *rq;
    spinlock_t queue_lock;
};
@@ -85,26 +83,18 @@ static int mtd_blktrans_thread(void *arg)
    current->flags |= PF_MEMALLOC | PF_NOFREEZE;

    spin_lock_irq(rq->queue_lock);
-
- while (!tr->blkcore_priv->exiting) {
+ while (!kthread_should_stop()) {
        struct request *req;
        struct mtd_blktrans_dev *dev;
        int res = 0;
- DECLARE_WAITQUEUE(wait, current);

        req = elv_next_request(rq);

        if (!req) {
- add_wait_queue(&tr->blkcore_priv->thread_wq, &wait);
- set_current_state(TASK_INTERRUPTIBLE);
-
```

```

spin_unlock_irq(rq->queue_lock);
-
+ set_current_state(TASK_INTERRUPTIBLE);
schedule();
- remove_wait_queue(&tr->blkcore_priv->thread_wq, &wait);
-
spin_lock_irq(rq->queue_lock);
-
continue;
}

@@ -123,13 +113,13 @@ static int mtd_blktrans_thread(void *arg)
}
spin_unlock_irq(rq->queue_lock);

- complete_and_exit(&tr->blkcore_priv->thread_dead, 0);
+ return 0;
}

static void mtd_blktrans_request(struct request_queue *rq)
{
    struct mtd_blktrans_ops *tr = rq->queuedata;
- wake_up(&tr->blkcore_priv->thread_wq);
+ wake_up_process(tr->blkcore_priv->thread);
}

@@ -355,7 +345,6 @@ static struct mtd_notifier blktrans_notifier = {

int register_mtd_blktrans(struct mtd_blktrans_ops *tr)
{
- struct task_struct *task;
int ret, i;

/* Register the notifier if/when the first device type is
@@ -379,8 +368,6 @@ int register_mtd_blktrans(struct mtd_blktrans_ops *tr)
    return ret;
}
spin_lock_init(&tr->blkcore_priv->queue_lock);
- init_completion(&tr->blkcore_priv->thread_dead);
- init_waitqueue_head(&tr->blkcore_priv->thread_wq);

tr->blkcore_priv->rq = blk_init_queue(mtd_blktrans_request, &tr->blkcore_priv->queue_lock);
if (!tr->blkcore_priv->rq) {
@@ -394,13 +381,14 @@ int register_mtd_blktrans(struct mtd_blktrans_ops *tr)
    blk_queue_hardsect_size(tr->blkcore_priv->rq, tr->blksize);
    tr->blkshift = ffs(tr->blksize) - 1;

```

```

- task = kthread_run(mtd_blktrans_thread, tr, "%sd", tr->name);
- if (IS_ERR(task)) {
+ tr->blkcore_priv->thread = kthread_run(mtd_blktrans_thread, tr,
+ "%sd", tr->name);
+ if (IS_ERR(tr->blkcore_priv->thread)) {
    blk_cleanup_queue(tr->blkcore_priv->rq);
    unregister_blkdev(tr->major, tr->name);
    kfree(tr->blkcore_priv);
    mutex_unlock(&mtd_table_mutex);
- return PTR_ERR(task);
+ return PTR_ERR(tr->blkcore_priv->thread);
}

INIT_LIST_HEAD(&tr->devs);
@@ -423,9 +411,7 @@ int deregister_mtd_blktrans(struct mtd_blktrans_ops *tr)
mutex_lock(&mtd_table_mutex);

/* Clean up the kernel thread */
- tr->blkcore_priv->exiting = 1;
- wake_up(&tr->blkcore_priv->thread_wq);
- wait_for_completion(&tr->blkcore_priv->thread_dead);
+ kthread_stop(tr->blkcore_priv->thread);

/* Remove it from the list of active majors */
list_del(&tr->list);

```

--  
dwmw2

---

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

---