
Subject: [PATCH] Fix SAK_work workqueue initialization.
Posted by [ebiederm](#) on Tue, 13 Feb 2007 21:38:58 GMT
[View Forum Message](#) <> [Reply to Message](#)

Somewhere in the rewrite of the work queues my cleanup of SAK handling got broken. Maybe I didn't retest it properly or possibly the API was changing so fast I missed something. Regardless currently triggering a SAK now generates an ugly BUG_ON and kills the kernel.

Thanks to Alexey Dobriyan <adobriyan@openvz.org> for spotting this.

This modifies the use of SAK_work to initialize it when the data structure it resides in is initialized, and to simply call schedule_work when we need to generate a SAK. I update both data structures that have a SAK_work member for consistency.

All of the old PREPARE_WORK calls that are now gone.

If we call schedule_work again before it has processed it has generated the first SAK it will simply ignore the duplicate schedule_work request.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

```
drivers/char/keyboard.c | 1 -
drivers/char/sysrq.c    | 1 -
drivers/char/tty_io.c   | 3 +--
drivers/char/vt.c       | 1 +
4 files changed, 2 insertions(+), 4 deletions(-)
```

```
diff --git a/drivers/char/keyboard.c b/drivers/char/keyboard.c
index c654a3e..cb8d691 100644
--- a/drivers/char/keyboard.c
+++ b/drivers/char/keyboard.c
@@ -596,7 +596,6 @@ static void fn_spawn_con(struct vc_data *vc)
 static void fn_SAK(struct vc_data *vc)
 {
     struct work_struct *SAK_work = &vc_cons[fg_console].SAK_work;
- PREPARE_WORK(SAK_work, vc_SAK);
     schedule_work(SAK_work);
 }
```

```
diff --git a/drivers/char/sysrq.c b/drivers/char/sysrq.c
index 3757610..be73c80 100644
--- a/drivers/char/sysrq.c
+++ b/drivers/char/sysrq.c
@@ -89,7 +89,6 @@ static struct sysrq_key_op sysrq_loglevel_op = {
 static void sysrq_handle_SAK(int key, struct tty_struct *tty)
```

```

{
    struct work_struct *SAK_work = &vc_cons[fg_console].SAK_work;
- PREPARE_WORK(SAK_work, vc_SAK);
    schedule_work(SAK_work);
}
static struct sysrq_key_op sysrq_SAK_op = {
diff --git a/drivers/char/tty_io.c b/drivers/char/tty_io.c
index 65672c5..5289254 100644
--- a/drivers/char/tty_io.c
+++ b/drivers/char/tty_io.c
@@ -3442,7 +3442,6 @@ void do_SAK(struct tty_struct *tty)
{
    if (!tty)
        return;
- PREPARE_WORK(&tty->SAK_work, do_SAK_work);
    schedule_work(&tty->SAK_work);
}

@@ -3568,7 +3567,7 @@ static void initialize_tty_struct(struct tty_struct *tty)
    mutex_init(&tty->atomic_write_lock);
    spin_lock_init(&tty->read_lock);
    INIT_LIST_HEAD(&tty->tty_files);
- INIT_WORK(&tty->SAK_work, NULL);
+ INIT_WORK(&tty->SAK_work, do_SAK_work);
}

/*
diff --git a/drivers/char/vt.c b/drivers/char/vt.c
index 94ce3e7..c3f8e38 100644
--- a/drivers/char/vt.c
+++ b/drivers/char/vt.c
@@ -2635,6 +2635,7 @@ static int __init con_init(void)
*/
for (currcons = 0; currcons < MIN_NR_CONSOLES; currcons++) {
    vc_cons[currcons].d = vc = alloc_bootmem(sizeof(struct vc_data));
+ INIT_WORK(&vc_cons[currcons].SAK_work, vc_SAK);
    visual_init(vc, currcons, 1);
    vc->vc_screenbuf = (unsigned short *)alloc_bootmem(vc->vc_screenbuf_size);
    vc->vc_kmalloced = 0;
--
1.4.4.1.g278f

```
