Subject: Alt+SysRq+K broken
Posted by adobriyan on Tue, 13 Feb 2007 14:40:59 GMT
View Forum Message <> Reply to Message

kernel BUG at kernel/workqueue.c:212! 100% reproducable

queue_work
__handle_sysrq
kbd_event
add_timer_randomness
kbd_event
input_event
atkbd_interrupt
serio_interrupt
i8042_interrupt
...

reverting the following patch make it work again.

commit 8b6312f4dcc1efe7975731b6c47dd134282bd9ac
Author: Eric W. Biederman <ebiederm@xmission.com>
Date: Sat Feb 10 01:44:34 2007 -0800

[PATCH] vt: refactor console SAK processing

This does several things.
- It moves looking up of the current foreground console into process context where we can safely take the semaphore that protects this operation.
- It uses the new flavor of work queue processing.
- This generates a factor of do_SAK, __do_SAK that runs immediately.
- This calls __do_SAK with the console semaphore held ensuring nothing else happens to the console while we process the SAK operation.
- With the console SAK processing moved into process context this patch removes the xchg operations that I used to attempt to attomically update struct pid, because of the strange locking used in the SAK processing. With SAK using the normal console semaphore nothing special is needed.

Cc: Oleg Nesterov <oleg@tv-sign.ru>
Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
Signed-off-by: Andrew Morton <akpm@linux-foundation.org>
Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>

diff --git a/drivers/char/keyboard.c b/drivers/char/keyboard.c
index 7a6c1c0..c654a3e 100644
--- a/drivers/char/keyboard.c
+++ b/drivers/char/keyboard.c

```
@@ -595,15 +595,9 @@ static void fn_spawn_con(struct vc_data

 static void fn_SAK(struct vc_data *vc)
 {
- struct tty_struct *tty = vc->vc_tty;
-
- /*
-  * SAK should also work in all raw modes and reset
-  * them properly.
-  */
- if (tty)
-  do_SAK(tty);
- reset_vc(vc);
+ struct work_struct *SAK_work = &vc_cons[fg_console].SAK_work;
+ PREPARE_WORK(SAK_work, vc_SAK);
+ schedule_work(SAK_work);
 }

 static void fn_null(struct vc_data *vc)
diff --git a/drivers/char/sysrq.c b/drivers/char/sysrq.c
index 7fd3cd5..3757610 100644
--- a/drivers/char/sysrq.c
+++ b/drivers/char/sysrq.c
@@ -88,9 +88,9 @@ static struct sysrq_key_op sysrq_logleve
 #ifdef CONFIG_VT
 static void sysrq_handle_SAK(int key, struct tty_struct *tty)
 {
- if (tty)
-  do_SAK(tty);
- reset_vc(vc_cons[fg_console].d);
+ 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 = {
  .handler = sysrq_handle_SAK,
diff --git a/drivers/char/tty_io.c b/drivers/char/tty_io.c
index 47a6eac..c57b1f4 100644
--- a/drivers/char/tty_io.c
+++ b/drivers/char/tty_io.c
@@ -3324,10 +3324,8 @@ #endif
  * Nasty bug: do_SAK is being called in interrupt context.  This can
  * deadlock.  We punt it up to process context.  AKPM - 16Mar2001
  */
-static void __do_SAK(struct work_struct *work)
+void __do_SAK(struct tty_struct *tty)
 {
- struct tty_struct *tty =
```

```
-  container_of(work, struct tty_struct, SAK_work);
 #ifdef TTY_SOFT_SAK
  tty_hangup(tty);
 #else
@@ -3394,6 +3392,13 @@ #else
 #endif
 }

+static void do_SAK_work(struct work_struct *work)
+{
+ struct tty_struct *tty =
+  container_of(work, struct tty_struct, SAK_work);
+ __do_SAK(tty);
+}
+
 /*
  * The tq handling here is a little racy - tty->SAK_work may already be queued.
  * Fortunately we don't need to worry, because if ->SAK_work is already queued,
@@ -3404,7 +3409,7 @@ void do_SAK(struct tty_struct *tty)
 {
  if (!tty)
   return;
- PREPARE_WORK(&tty->SAK_work, __do_SAK);
+ PREPARE_WORK(&tty->SAK_work, do_SAK_work);
  schedule_work(&tty->SAK_work);
 }

diff --git a/drivers/char/vt_ioctl.c b/drivers/char/vt_ioctl.c
index dc8368e..3a5d301 100644
--- a/drivers/char/vt_ioctl.c
+++ b/drivers/char/vt_ioctl.c
@@ -672,7 +672,8 @@ #endif
   vc->vt_mode = tmp;
   /* the frsig is ignored, so we set it to 0 */
   vc->vt_mode.frsig = 0;
-  put_pid(xchg(&vc->vt_pid, get_pid(task_pid(current))));
+  put_pid(vc->vt_pid);
+  vc->vt_pid = get_pid(task_pid(current));
   /* no switch is required -- saw@shade.msu.ru */
   vc->vt_newvt = -1;
   release_console_sem();
@@ -1063,12 +1064,35 @@ void reset_vc(struct vc_data *vc)
  vc->vt_mode.relsig = 0;
  vc->vt_mode.acqsig = 0;
  vc->vt_mode.frsig = 0;
- put_pid(xchg(&vc->vt_pid, NULL));
+ put_pid(vc->vt_pid);
+ vc->vt_pid = NULL;
```

```
  vc->vt_newvt = -1;
  if (!in_interrupt())    /* Via keyboard.c:SAK() - akpm */
   reset_palette(vc);
 }

+void vc_SAK(struct work_struct *work)
+{
+ struct vc *vc_con =
+  container_of(work, struct vc, SAK_work);
+ struct vc_data *vc;
+ struct tty_struct *tty;
+
+ acquire_console_sem();
+ vc = vc_con->d;
+ if (vc) {
+  tty = vc->vc_tty;
+  /*
+   * SAK should also work in all raw modes and reset
+   * them properly.
+   */
+  if (tty)
+   __do_SAK(tty);
+  reset_vc(vc);
+ }
+ release_console_sem();
+}
+
 /*
  * Performs the back end of a vt switch
  */
diff --git a/include/linux/console_struct.h b/include/linux/console_struct.h
index ed6c0fe..a86162b 100644
--- a/include/linux/console_struct.h
+++ b/include/linux/console_struct.h
@@ -11,6 +11,7 @@

 #include <linux/wait.h>
 #include <linux/vt.h>
+#include <linux/workqueue.h>

 struct vt_struct;

@@ -103,6 +104,7 @@ struct vc_data {

 struct vc {
  struct vc_data *d;
+ struct work_struct SAK_work;
```

```
 /* might add  scrmem, vt_struct, kbd  at some time,
    to have everything in one place - the disadvantage
@@ -110,6 +112,7 @@ struct vc {
 };

 extern struct vc vc_cons [MAX_NR_CONSOLES];
+extern void vc_SAK(struct work_struct *work);

 #define CUR_DEF  0
 #define CUR_NONE 1
diff --git a/include/linux/tty.h b/include/linux/tty.h
index 65cbcf2..0161a8c 100644
--- a/include/linux/tty.h
+++ b/include/linux/tty.h
@@ -291,6 +291,7 @@ extern void tty_vhangup(struct tty_struc
 extern void tty_unhangup(struct file *filp);
 extern int tty_hung_up_p(struct file * filp);
 extern void do_SAK(struct tty_struct *tty);
+extern void __do_SAK(struct tty_struct *tty);
 extern void disassociate_ctty(int priv);
 extern void tty_flip_buffer_push(struct tty_struct *tty);
 extern speed_t tty_get_baud_rate(struct tty_struct *tty);
```