Subject: Re: [PATCH] vt: Make SAK run in process context.
Posted by ebiederm on Mon, 11 Dec 2006 21:27:40 GMT
View Forum Message <> Reply to Message

Andrew Morton <akpm@osdl.org> writes:

```
> On Mon, 11 Dec 2006 06:07:03 -0700
> ebiederm@xmission.com (Eric W. Biederman) wrote:
>
>>
>> This defers SAK so we can use the normal console semaphore to order
>> operations.
>>
>> This removes the xchg operations that I used to attempt to attmically
>> update struct pid, because of the strange locking used for SAK.  With
>> SAK using the normal console semaphore nothing special is needed.
>>
>
> This is all a bit smelly.
```

Ok. I will take a second look, thanks for catching this.

I think I was half blind when I prepared this patch, I missed
that do_SAK was scheduling work itself.

```
>>
>> +void deferred_SAK(void *data)
>> +{
>> + struct vc *vc_con = data;
>> + 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();
>> +}
>
> And a workqueue callback which calls a function which immediately does
```

> another schedule_work().
>
> I suspect you can fix all of this by passing a function pointer into
> do_SAK(): to either __do_SAK or to some new function which does the vc
> lookup then calls __do_SAK().

Yes. It looks like all I need is an appropriate factor of __do_SAK() that
I can call immediately.

> It probably means that you'll need to pass some payload into the workqueue
> callback, and dhowells just went and broke that on us. That can be fixed
> by adding a new `void *tty_struct.SAK_work_data'.
>
>
> hmm, do_SAK() is being a bit bad, overwriting the ->SAK_work on a
> work_struct which might presently be scheduled. To do this safely we need
> a new variant on queue_work():

And of course there is the truly silly issue that X spells uses
Ctrl-Alt-Backspace instead of the kernel provided SAK to implement this.

Regardless that looks right. Unless there is some locking on the tty we
can exploit.

```
> int queue_work_with_data(struct workqueue_struct *wq,
>    struct work_struct *work, void **datap, void *data
> {
>  int ret = 0, cpu = get_cpu();
>
>  if (!test_and_set_bit(WORK_STRUCT_PENDING, &work->management)) {
>   if (datap)
>    *datap = data;
>   if (unlikely(is_single_threaded(wq)))
>    cpu = singlethread_cpu;
>   BUG_ON(!list_empty(&work->entry));
>   __queue_work(per_cpu_ptr(wq->cpu_wq, cpu), work);
>   ret = 1;
>  }
>  put_cpu();
>  return ret;
> }
>
> then, of course,
>
> int queue_work(struct workqueue_struct *wq, struct work_struct *work)
> {
>  return queue_work_with_data(wq, work, NULL, NULL);
> }
```

>
> iirc, other places in the kernel need queue_work_with_data(), for removal
> of the *_WORK_NAR() stuff.

Wow. The intersection of the clean ups.

Eric