Subject: Re: [PATCH v5] posix timers: allocate timer id per process
Posted by Thomas Gleixner on Tue, 23 Oct 2012 09:50:11 GMT
View Forum Message <> Reply to Message

B1;2601;0cOn Tue, 23 Oct 2012, Stanislav Kinsbursky wrote:
> Patch replaces global idr with global hash table for posix timers and
> makes timer ids unique not globally, but per process. Next free timer id is
> type of integer and stored on signal struct (posix_timer_id). If free timer id
> reaches negative value on timer creation, it will be dropped to zero and
> -EAGAIN will be returned to user.

That's the theory ...

> diff --git a/include/linux/sched.h b/include/linux/sched.h
> index 0dd42a0..dce1651 100644
> --- a/include/linux/sched.h
> +++ b/include/linux/sched.h
> @@ -51,6 +51,7 @@ struct sched_param {
> #include <linux/cred.h>
> #include <linux/llist.h>
> #include <linux/uidgid.h>
> +#include <linux/idr.h>

Why ?

> +static int posix_timer_add(struct k_itimer *timer)
> +{
> + struct signal_struct *sig = current->signal;
> + int next_free_id = sig->posix_timer_id;
> + struct hlist_head *head;
> + int ret = -ENOENT;
> +
> + do {
> + spin_lock(&hash_lock);
> + head = &posix_timers_hashtable[hash(sig, sig->posix_timer_id)];
> + if (!__posix_timers_find(head, sig, sig->posix_timer_id)) {
> +   hlist_add_head_rcu(&timer->t_hash, head);
> +   ret = sig->posix_timer_id++;

Let's assume a program, which creates timers and destroys them in a
loop.

```
while (1) {
      id = timer_create();
      if (id < 0)
        continue;
      timer_delete(id);
}
```

After 2^31 iterations sig->posix_timer_id contains 0x80000000.

__posix_timer_find() will return NULL as there is no timer with this id and you happily add the new timer to the hash list and return 0x80000000, which translates to -INT_MAX.

Now this will return a totally useless error code to user space and what's worse it will free that timer without removing it from the hash bucket. The next access to that bucket will explode nicely.

```
> +  } else {
> +   if (++sig->posix_timer_id < 0)
> +    sig->posix_timer_id = 0;
> +   if (sig->posix_timer_id == next_free_id)
> +    ret = -EAGAIN;
```

This code path has obvioulsy never been executed.

```
> +  }
> +  spin_unlock(&hash_lock);
> + } while (ret == -ENOENT);
> + return ret;
> +}
```

Thanks,

tglx