Subject: Re: [PATCH 2/3] user.c: use kmem_cache_zalloc()
Posted by Satyam Sharma on Sat, 22 Sep 2007 04:30:29 GMT
View Forum Message <> Reply to Message

On Fri, 21 Sep 2007, Andrew Morton wrote:

```
>
> On Fri, 21 Sep 2007 13:39:06 +0400
> Alexey Dobriyan <adobriyan@sw.ru> wrote:
>
> > Quite a few fields are zeroed during user_struct creation, so use
> > kmem_cache_zalloc() --  save a few lines and #ifdef. Also will help avoid
> >  #ifdef CONFIG_POSIX_MQUEUE in next patch.
> >
> > Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>
> > ---
> >
> >  kernel/user.c |   13 +------------
> >  1 file changed, 1 insertion(+), 12 deletions(-)
> >
> > --- a/kernel/user.c
> > +++ b/kernel/user.c
> > @@ -129,21 +129,11 @@ struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
> >   if (!up) {
> >    struct user_struct *new;
> >
> > -  new = kmem_cache_alloc(uid_cachep, GFP_KERNEL);
> > +  new = kmem_cache_zalloc(uid_cachep, GFP_KERNEL);
> >    if (!new)
> >     return NULL;
> >    new->uid = uid;
> >    atomic_set(&new->__count, 1);
> > -  atomic_set(&new->processes, 0);
> > -  atomic_set(&new->files, 0);
> > -  atomic_set(&new->sigpending, 0);
> > -#ifdef CONFIG_INOTIFY_USER
> > -  atomic_set(&new->inotify_watches, 0);
> > -  atomic_set(&new->inotify_devs, 0);
> > -#endif
> > -
> > -  new->mq_bytes = 0;
> > -  new->locked_shm = 0;
>
>
> This assumes that setting an atomic_t to the all-zeroes pattern is
> equivalent to atomic_set(v, 0).
>
> This happens to be true for all present architectures, afaik.  But an
```

> architecture which has crappy primitives could quite legitimately implement
> its atomic_t as:
>
> typedef struct {
> int counter;
> spinlock_t lock;
> } atomic_t;
>
> in which case your assumption breaks.

Agreed, and this (implementing atomic ops using spinlocks) is already true for the CRIS platform.

However, cris' implementation explicitly takes care to ensure that atomic_t contains just a solitary int member, and no spinlock_t's inside the atomic_t itself. [include/asm-cris/arch-v32/atomic.h]

Of course, that "128" limits scalability, so no more than 128 CPUs can be executing atomic ops at any given instant of time, but admittedly I'm getting anal here myself ... (but probably that's often perfectly the right attitude to have too)

> So it's all a bit theoretical and a bit anal, and I'm sure we're making the
> same mistake in other places, but it's not a change I particularly like..

Hmm, it's borderline.

Such changes make text smaller (in terms of LOC as well vmlinux size).

But they also hurt grepping. Often we (at least I) want to grep for when is a variable/struct member/etc getting initialized or getting set/assigned to. Take this case, for example -- I bet it's important (for overall logic) that those variables get initialized to zero. But *zalloc() functions do that implicitly, so it wastes precious seconds or minutes of developer time when grepping that code.

OTOH, we could make it standard practise to put a little comment on top of such *zalloc() usages, explicitly enumerating the struct members that that the *zalloc() is assumed to initialize to zero.

<runs away>

Satyam