Subject: Re: [RFC][PATCH] rename 'struct pid'
Posted by Dave Hansen on Wed, 11 Apr 2007 18:11:32 GMT

On Wed, 2007-04-11 at 11:34 -0600, Eric W. Biederman wrote:
> A struct pid is very similar to a list symbol. Two of them can be compared for
> equality just by comparing pointers. You can get different information about
> a struct pid by examining different "slots". A struct pid has a
> name, but unlike lisp the name is numeric instead of a string.

I'm not sure what you mean by "list symbol". Is that a lisp concept?

> Furthermore there is a one to one mapping from all valid user space
> pid numbers (in a single namespace) to a kernel struct pid. Whether
> you compare the pointer is kernel space or the numeric user space
> value you will get the same answer.
>
> The kernel struct pid can however live on after it's numeric name is
> no longer visible to user space.
>
> The kernel struct pid is an identifier even if it isn't numeric.
>
> The point of struct pid is so you can operate on pids without having
> to worry about the pesky user space numeric names.

Userspace refers to processes, sessions, and process groups by numbers
which are finite and can wrap over time. Instead of using these numbers
inside the kernel, we use a 'struct pid'. Unlike a plain number, a
'struct pid' uniquely refers to a particular process, session, or
process group, and does not suffer from any wrapping effects.

In general, the underlying task to which a 'struct pid' refers will not
change. It is possible such as when a non-thread-leader does an exec()
and takes over as the leader, tat the tasks to which a 'struct pid'
refers will change.

This effectively lets the kernel do a pid_t (process, session, or
process group) to task lookup at a particular time and keep the results
of that lookup meaningful for a long, long time, not worrying about if
userspace has re-used those values.

I think at least part of the problem is that a 'struct pid' creates a
relationship to pid, as well as session ids and process groups. This is
a bit muddled relationship that comes out of userspace, but it would be
nice to unmuddle it somehow in the kernel. I think part of that might
be to give 'struct pid' a more neutral name that also encompasses the

pgrp and sid parts.

'pref' seems a decent compromise. It keeps the "process" notion around (and thus connected to the userspace concepts) while helping to reduce the use of 'pid' in the name.

These can be a bit confusing:

```
 struct pid *pid;
 struct pid *pgrp;
 struct pid *sid;
```

this is a bit better:

```
 struct pref *pid;
 struct pref *pgrp;
 struct pref *sid;
```

-- Dave