Subject: Re: [PATCH] Signal semantics for /sbin/init
Posted by Dave Hansen on Mon, 29 Oct 2007 16:13:37 GMT
View Forum Message <> Reply to Message

On Sat, 2007-10-27 at 12:00 -0700, sukadev@us.ibm.com wrote:
> +static int sig_init_ignore(struct task_struct *tsk)
> +{
> -static int sig_ignored(struct task_struct *t, int sig)
> + // Currently this check is a bit racy with exec(),
> + // we can _simplify_ de_thread and close the race.
> + if (likely(!is_global_init(tsk->group_leader)))
> + return 0;
> +
> + return 1;
> +}

Umm. C++ comments? ;)

Also, I'm not sure an out-of-line function this small really needs a likely/unlikely hint. Seems like a waste of the bytes to me.

> +static int sig_task_ignore(struct task_struct *tsk, int sig)
> {
> - void __user * handler;
> + void __user * handler = tsk->sighand->action[sig-1].sa.sa_handler;
> +
> + if (handler == SIG_IGN)
> + return 1;

Something simple like:
/*
* ensure that the process is expecting to get this signal,
* and thus won't get SIG_...
*/
Would be helpful to understand your motivation here.

> + if (handler != SIG_DFL)
> + return 0;
>
> + return sig_kernel_ignore(sig) || sig_init_ignore(tsk);
> +}

> +static int sig_ignored(struct task_struct *t, int sig)
> +{
> /*
> * Tracers always want to know about signals..
> */
> @@ -58,10 +80,7 @@ static int sig_ignored(struct task_struc

>   if (sigismember(&t->blocked, sig))
>   return 0;
>
> - /* Is it explicitly or implicitly ignored? */
> - handler = t->sighand->action[sig-1].sa.sa_handler;
> - return   handler == SIG_IGN ||
> - (handler == SIG_DFL && sig_kernel_ignore(sig));
> + return sig_task_ignore(t, sig);
> }

And technically, your helper patch could be separated out from the rest here. It isn't a huge deal, but it would be nice.

> /*
> @@ -566,6 +585,9 @@ static void handle_stop_signal(int sig,
>   */
>   return;
>
> + if (sig_init_ignore(p))
> + return;

The function name isn't quite good enough to understand why this is here. Quick comment, maybe?

>   if (sig_kernel_stop(sig)) {
>   /*
>   * This is a stop signal. Remove SIGCONT from all queues.
> @@ -1860,12 +1882,6 @@ relock:
>   if (sig_kernel_ignore(signr)) /* Default is nothing. */
>   continue;
>
> - /*
> -   * Global init gets no signals it doesn't want.
> -   */
> - if (is_global_init(current))
> -   continue;
> -
>   if (sig_kernel_stop(signr)) {
>   /*
>   * The default action is to stop all threads in
> @@ -2317,6 +2333,7 @@ int do_sigaction(int sig, struct k_sigac
>   k = &current->sighand->action[sig-1];
>
>   spin_lock_irq(&current->sighand->siglock);
> +
>   if (oact)
>   *oact = *k;

Fix that, please.

> @@ -2335,8 +2352,7 @@ int do_sigaction(int sig, struct k_sigac
> * (for example, SIGCHLD), shall cause the pending signal to
> * be discarded, whether or not it is blocked"
> */
> - if (act->sa.sa_handler == SIG_IGN ||
> - (act->sa.sa_handler == SIG_DFL && sig_kernel_ignore(sig))) {
> + if (sig_task_ignore(current, sig)) {
> struct task_struct *t = current;
> sigemptyset(&mask);
> sigaddset(&mask, sig);

-- Dave

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers