Subject: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Pavel Emelianov on Wed, 16 Jan 2008 12:58:55 GMT
View Forum Message <> Reply to Message

There's only one bit in the clone_flags left, so we won't be able
to create more namespaces after we make it busy. Besides, for
checkpoint/restart jobs we might want to create tasks with given
pids (virtual of course). And nobody knows for sure what else might
be required from clone() in the future.

This is an attempt to create a extendable API for clone and unshare.
Actually this patch is a request for comment about the overall
design. If it will turn out to "look good", then we'll select some
better names for new flag and data types.

I use the last bit in the clone_flags for CLONE_LONGARG. When set it
will denote that the child_tidptr is not a pointer to a tid storage,
but the pointer to the struct long_clone_struct which currently
looks like this:

struct long_clone_arg {
 int size;
};

When we want to add a new argument for clone we just put it *at the
end of this structure* and adjust the size. The binary compatibility
with older long_clone_arg-s is facilitated with the helper macro
clone_arg_has(). It checks, that the desired member is provided from
the userspace.

Cedric implemented the same thing for unshare - he added the second
argument for it and iff the CLONE_NEWCLONE is specified - uses it.
Binary compatibility with the old ushare will be kept.

The new argument is to be pulled up to the create_new_namespaces,
which is done in a second patch.

This patch is made against the 2.6.24-rc6-mm1. It is tested to pass
more flags into create_new_namespaces.

Signed-off-by: Pavel Emelyanov <xemul@openzv.org>
Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Reviewed-by: Serge Hallyn <serue@us.ibm.com>

---

diff --git a/include/linux/sched.h b/include/linux/sched.h
index 0a15018..f9332bb 100644

```
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -27,6 +27,23 @@
 #define CLONE_NEWUSER  0x10000000 /* New user namespace */
 #define CLONE_NEWPID  0x20000000 /* New pid namespace */
 #define CLONE_NEWNET  0x40000000 /* New network namespace */
+#define CLONE_LONGARG  0x80000000 /* Has an long_clone_arg */
+
+/*
+ * If the CLONE_LONGARG argument is specified, then the
+ * child_tidptr is expected to point to the struct long_clone_arg.
+ *
+ * This structure has a variable size, which is to be recorded
+ * in the size member. All other members a to be put after this.
+ * Never ... No. Always add new members only at its tail.
+ *
+ * The clone_arg_has() macro is responsible for checking whether
+ * the given arg has the desired member.
+ */
+
+struct long_clone_arg {
+ int size;
+};

 /*
  * Scheduling policies
@@ -40,6 +57,11 @@

 #ifdef __KERNEL__

+#define clone_arg_has(arg, member) ({ \
+  struct long_clone_arg *__carg = arg; \
+  (__carg->size >= offsetof(struct long_clone_arg, member) + \
+     sizeof(__carg->member)); })
+
 struct sched_param {
  int sched_priority;
 };
diff --git a/include/linux/syscalls.h b/include/linux/syscalls.h
index 4c2577b..1cf3541 100644
--- a/include/linux/syscalls.h
+++ b/include/linux/syscalls.h
@@ -585,7 +585,7 @@ asmlinkage long compat_sys_newfstatat(unsigned int dfd, char __user *
filename,
           int flag);
 asmlinkage long compat_sys_openat(unsigned int dfd, const char __user *filename,
        int flags, int mode);
-asmlinkage long sys_unshare(unsigned long unshare_flags);
```

```
+asmlinkage long sys_unshare(unsigned long unshare_flags, int __user *flagptr);

 asmlinkage long sys_splice(int fd_in, loff_t __user *off_in,
       int fd_out, loff_t __user *off_out,
diff --git a/kernel/fork.c b/kernel/fork.c
index 19873c7..5e85567 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -967,6 +967,33 @@ static void rt_mutex_init_task(struct task_struct *p)
 #endif
 }

+static struct long_clone_arg *get_long_clone_arg(unsigned long flags,
+  int __user *child_tidptr)
+{
+ int size;
+ struct long_clone_arg *carg;
+
+ if (!(flags & CLONE_LONGARG))
+  return NULL;
+
+ if (get_user(size, child_tidptr))
+  return ERR_PTR(-EFAULT);
+
+ if (size > sizeof(struct long_clone_arg))
+  return ERR_PTR(-EINVAL);
+
+ carg = kzalloc(sizeof(struct long_clone_arg), GFP_KERNEL);
+ if (carg == NULL)
+  return ERR_PTR(-ENOMEM);
+
+ if (copy_from_user(carg, child_tidptr, size)) {
+  kfree(carg);
+  return ERR_PTR(-EFAULT);
+ }
+
+ return carg;
+}
+
 /*
  * This creates a new process as a copy of the old one,
  * but does not actually start it yet.
@@ -985,6 +1012,7 @@ static struct task_struct *copy_process(unsigned long clone_flags,
  int retval;
  struct task_struct *p;
  int cgroup_callbacks_done = 0;
+ struct long_clone_arg *carg;
```

```
  if ((clone_flags & (CLONE_NEWNS|CLONE_FS)) == (CLONE_NEWNS|CLONE_FS))
   return ERR_PTR(-EINVAL);
@@ -1004,6 +1032,11 @@ static struct task_struct *copy_process(unsigned long clone_flags,
  if ((clone_flags & CLONE_SIGHAND) && !(clone_flags & CLONE_VM))
   return ERR_PTR(-EINVAL);

+ if ((clone_flags & CLONE_LONGARG) &&
+    (clone_flags & (CLONE_CHILD_SETTID |
+      CLONE_CHILD_CLEARTID)))
+  return ERR_PTR(-EINVAL);
+
  retval = security_task_create(clone_flags);
  if (retval)
   goto fork_out;
@@ -1141,8 +1174,14 @@ static struct task_struct *copy_process(unsigned long clone_flags,
  /* Perform scheduler related setup. Assign this task to a CPU. */
  sched_fork(p, clone_flags);

- if ((retval = security_task_alloc(p)))
+ carg = get_long_clone_arg(clone_flags, child_tidptr);
+ if (IS_ERR(carg)) {
+  retval = PTR_ERR(carg);
   goto bad_fork_cleanup_policy;
+ }
+
+ if ((retval = security_task_alloc(p)))
+  goto bad_fork_cleanup_carg;
  if ((retval = audit_alloc(p)))
   goto bad_fork_cleanup_security;
  /* copy all the process information */
@@ -1355,6 +1394,8 @@ bad_fork_cleanup_audit:
  audit_free(p);
 bad_fork_cleanup_security:
  security_task_free(p);
+bad_fork_cleanup_carg:
+ kfree(carg);
 bad_fork_cleanup_policy:
 #ifdef CONFIG_NUMA
  mpol_free(p->mempolicy);
@@ -1672,7 +1713,7 @@ static int unshare_semundo(unsigned long unshare_flags, struct
sem_undo_list **n
  * constructed. Here we are modifying the current, active,
  * task_struct.
  */
-asmlinkage long sys_unshare(unsigned long unshare_flags)
+asmlinkage long sys_unshare(unsigned long unshare_flags, int __user *flagptr)
 {
  int err = 0;
```

```
  struct fs_struct *fs, *new_fs = NULL;
@@ -1681,6 +1722,7 @@ asmlinkage long sys_unshare(unsigned long unshare_flags)
  struct files_struct *fd, *new_fd = NULL;
  struct sem_undo_list *new_ulist = NULL;
  struct nsproxy *new_nsproxy = NULL;
+ struct long_clone_arg *carg;

  check_unshare_flags(&unshare_flags);

@@ -1689,11 +1731,17 @@ asmlinkage long sys_unshare(unsigned long unshare_flags)
  if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
     CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
     CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER|
-    CLONE_NEWNET))
+    CLONE_NEWNET|CLONE_LONGARG))
   goto bad_unshare_out;

- if ((err = unshare_thread(unshare_flags)))
+ carg = get_long_clone_arg(unshare_flags, flagptr);
+ if (IS_ERR(carg)) {
+  err = PTR_ERR(carg);
   goto bad_unshare_out;
+ }
+
+ if ((err = unshare_thread(unshare_flags)))
+  goto bad_unshare_cleanup_carg;
  if ((err = unshare_fs(unshare_flags, &new_fs)))
   goto bad_unshare_cleanup_thread;
  if ((err = unshare_sighand(unshare_flags, &new_sigh)))
@@ -1763,6 +1811,8 @@ bad_unshare_cleanup_fs:
   put_fs_struct(new_fs);

 bad_unshare_cleanup_thread:
+bad_unshare_cleanup_carg:
+ kfree(carg);
 bad_unshare_out:
  return err;
 }
```



---

Subject: [PATCH 2/2] Propagate the long_clone_arg up to the create_new_namespaces
Posted by Pavel Emelianov on Wed, 16 Jan 2008 13:00:57 GMT

The first user for the long_clone_arg is the namespaces code,
so pull the extended argument up to the namespaces creation
function.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Reviewed-by: Serge Hallyn <serue@us.ibm.com>

---

```
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
index 0e66b57..ad93b19 100644
--- a/include/linux/nsproxy.h
+++ b/include/linux/nsproxy.h
@@ -62,12 +62,13 @@ static inline struct nsproxy *task_nsproxy(struct task_struct *tsk)
  return rcu_dereference(tsk->nsproxy);
 }

-int copy_namespaces(unsigned long flags, struct task_struct *tsk);
+int copy_namespaces(unsigned long flags, struct task_struct *tsk,
+  struct long_clone_arg *carg);
 void exit_task_namespaces(struct task_struct *tsk);
 void switch_task_namespaces(struct task_struct *tsk, struct nsproxy *new);
 void free_nsproxy(struct nsproxy *ns);
 int unshare_nsproxy_namespaces(unsigned long, struct nsproxy **,
- struct fs_struct *);
+ struct fs_struct *, struct long_clone_arg *carg);

 static inline void put_nsproxy(struct nsproxy *ns)
 {
diff --git a/kernel/fork.c b/kernel/fork.c
index 19873c7..5e85567 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1160,7 +1199,7 @@ static struct task_struct *copy_process(unsigned long clone_flags,
   goto bad_fork_cleanup_signal;
  if ((retval = copy_keys(clone_flags, p)))
   goto bad_fork_cleanup_mm;
- if ((retval = copy_namespaces(clone_flags, p)))
+ if ((retval = copy_namespaces(clone_flags, p, carg)))
   goto bad_fork_cleanup_keys;
  retval = copy_thread(0, clone_flags, stack_start, stack_size, p, regs);
  if (retval)
@@ -1705,7 +1753,7 @@ asmlinkage long sys_unshare(unsigned long unshare_flags)
  if ((err = unshare_semundo(unshare_flags, &new_ulist)))
   goto bad_unshare_cleanup_fd;
  if ((err = unshare_nsproxy_namespaces(unshare_flags, &new_nsproxy,
```

```
-   new_fs)))
+   new_fs, carg)))
    goto bad_unshare_cleanup_semundo;

  if (new_fs ||  new_mm || new_fd || new_ulist || new_nsproxy) {
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index f5d332c..5fd4a03 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -48,7 +48,8 @@ static inline struct nsproxy *clone_nsproxy(struct nsproxy *orig)
  * leave it to the caller to do proper locking and attach it to task.
  */
 static struct nsproxy *create_new_namespaces(unsigned long flags,
-   struct task_struct *tsk, struct fs_struct *new_fs)
+   struct task_struct *tsk, struct fs_struct *new_fs,
+   struct long_clone_arg *carg)
 {
  struct nsproxy *new_nsp;
  int err;
@@ -119,7 +120,8 @@ out_ns:
  * called from clone.  This now handles copy for nsproxy and all
  * namespaces therein.
  */
-int copy_namespaces(unsigned long flags, struct task_struct *tsk)
+int copy_namespaces(unsigned long flags, struct task_struct *tsk,
+  struct long_clone_arg *carg)
 {
  struct nsproxy *old_ns = tsk->nsproxy;
  struct nsproxy *new_ns;
@@ -131,7 +133,8 @@ int copy_namespaces(unsigned long flags, struct task_struct *tsk)
  get_nsproxy(old_ns);

  if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-    CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET)))
+    CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET |
+    CLONE_LONGARG)))
    return 0;

  if (!capable(CAP_SYS_ADMIN)) {
@@ -139,7 +142,7 @@ int copy_namespaces(unsigned long flags, struct task_struct *tsk)
    goto out;
  }

- new_ns = create_new_namespaces(flags, tsk, tsk->fs);
+ new_ns = create_new_namespaces(flags, tsk, tsk->fs, carg);
  if (IS_ERR(new_ns)) {
   err = PTR_ERR(new_ns);
   goto out;
```

```
@@ -179,19 +182,20 @@ void free_nsproxy(struct nsproxy *ns)
  * On success, returns the new nsproxy.
  */
 int unshare_nsproxy_namespaces(unsigned long unshare_flags,
-  struct nsproxy **new_nsp, struct fs_struct *new_fs)
+  struct nsproxy **new_nsp, struct fs_struct *new_fs,
+  struct long_clone_arg *carg)
 {
  int err = 0;

  if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-         CLONE_NEWUSER | CLONE_NEWNET)))
+         CLONE_NEWUSER | CLONE_NEWNET | CLONE_LONGARG)))
   return 0;

  if (!capable(CAP_SYS_ADMIN))
   return -EPERM;

  *new_nsp = create_new_namespaces(unshare_flags, current,
-    new_fs ? new_fs : current->fs);
+    new_fs ? new_fs : current->fs, carg);
  if (IS_ERR(*new_nsp)) {
   err = PTR_ERR(*new_nsp);
   goto out;
```



---

## Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by corbet on Wed, 16 Jan 2008 14:23:40 GMT
View Forum Message <> Reply to Message

Hi, Pavel,

[Adding Ulrich]

> I use the last bit in the clone_flags for CLONE_LONGARG. When set it
> will denote that the child_tidptr is not a pointer to a tid storage,
> but the pointer to the struct long_clone_struct which currently
> looks like this:

I'm probably just totally off the deep end, but something did occur to me: this looks an awful lot like a special version of the sys_indirect() idea. Unless it has been somehow decided that sys_indirect() is the wrong idea, might it not be better to look at making that interface

solve the extended clone() problem as well?

jon

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

---

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Pavel Emelianov on Wed, 16 Jan 2008 15:05:30 GMT
View Forum Message <> Reply to Message

Jonathan Corbet wrote:
> Hi, Pavel,
>
> [Adding Ulrich]
>
>> I use the last bit in the clone_flags for CLONE_LONGARG. When set it
>> will denote that the child_tidptr is not a pointer to a tid storage,
>> but the pointer to the struct long_clone_struct which currently
>> looks like this:
>
> I'm probably just totally off the deep end, but something did occur to
> me: this looks an awful lot like a special version of the sys_indirect()
> idea.  Unless it has been somehow decided that sys_indirect() is the
> wrong idea, might it not be better to look at making that interface
> solve the extended clone() problem as well?

We has such an idea, but as far as I know sys_indirect idea is
not yet accomplished. So I risked to propose such an extension
for sys_clone.

If Andrew says that sys_indirect is on its way to -mm tree, then
we'll surely wait and use this one for namespaces.

> jon
>

Thanks,
Pavel

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

---

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Al Viro on Thu, 17 Jan 2008 03:48:44 GMT
View Forum Message <> Reply to Message

On Wed, Jan 16, 2008 at 07:23:40AM -0700, Jonathan Corbet wrote:
> Hi, Pavel,
>
> [Adding Ulrich]
>
> > I use the last bit in the clone_flags for CLONE_LONGARG. When set it
> > will denote that the child_tidptr is not a pointer to a tid storage,
> > but the pointer to the struct long_clone_struct which currently
> > looks like this:
>
> I'm probably just totally off the deep end, but something did occur to
> me: this looks an awful lot like a special version of the sys_indirect()
> idea. Unless it has been somehow decided that sys_indirect() is the
> wrong idea, might it not be better to look at making that interface
> solve the extended clone() problem as well?

Nah, just put an XML parser into the kernel to have the form match the
contents...

Al "perhaps we should newgroup alt.tasteless.api for all that stuff" Viro

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

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Cedric Le Goater on Thu, 17 Jan 2008 09:28:32 GMT
View Forum Message <> Reply to Message

Al Viro wrote:
> On Wed, Jan 16, 2008 at 07:23:40AM -0700, Jonathan Corbet wrote:
>> Hi, Pavel,
>>
>> [Adding Ulrich]
>>
>>> I use the last bit in the clone_flags for CLONE_LONGARG. When set it
>>> will denote that the child_tidptr is not a pointer to a tid storage,
>>> but the pointer to the struct long_clone_struct which currently
>>> looks like this:
>> I'm probably just totally off the deep end, but something did occur to
>> me: this looks an awful lot like a special version of the sys_indirect()
>> idea. Unless it has been somehow decided that sys_indirect() is the
>> wrong idea, might it not be better to look at making that interface

>> solve the extended clone() problem as well?
>
> Nah, just put an XML parser into the kernel to have the form match the
> contents...
>
> Al "perhaps we should newgroup alt.tasteless.api for all that stuff" Viro

so you'd rather have new syscalls to support new clone flags ? something
like :

 long sys_clone64(unsigned long flags_high, unsigned long flag_low)
 long sys_unshare64(unsigned long flags_high, unsigned long flag_low)

Thanks,

C.

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

---

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Pavel Machek on Wed, 23 Jan 2008 20:59:30 GMT
View Forum Message <> Reply to Message

On Wed 2008-01-16 15:58:55, Pavel Emelyanov wrote:
> There's only one bit in the clone_flags left, so we won't be able
> to create more namespaces after we make it busy. Besides, for
> checkpoint/restart jobs we might want to create tasks with given
> pids (virtual of course). And nobody knows for sure what else might
> be required from clone() in the future.
>
> This is an attempt to create a extendable API for clone and unshare.
> Actually this patch is a request for comment about the overall
> design. If it will turn out to "look good", then we'll select some
> better names for new flag and data types.
>
> I use the last bit in the clone_flags for CLONE_LONGARG. When set it
> will denote that the child_tidptr is not a pointer to a tid storage,
> but the pointer to the struct long_clone_struct which currently
> looks like this:
>
> struct long_clone_arg {
>  int size;
> };

Ugly as night, I'd say. (Al said it better). What about just adding

clone2 syscall, that takes u64?
--
(english) http://www.livejournal.com/~pavelmachek
(cesky, pictures) http://atrey.karlin.mff.cuni.cz/~pavel/picture/horses/blog.html

_____

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

---

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Cedric Le Goater on Thu, 24 Jan 2008 17:09:56 GMT
View Forum Message <> Reply to Message

Pavel Machek wrote:
> On Wed 2008-01-16 15:58:55, Pavel Emelyanov wrote:
>> There's only one bit in the clone_flags left, so we won't be able
>> to create more namespaces after we make it busy. Besides, for
>> checkpoint/restart jobs we might want to create tasks with given
>> pids (virtual of course). And nobody knows for sure what else might
>> be required from clone() in the future.
>>
>> This is an attempt to create a extendable API for clone and unshare.
>> Actually this patch is a request for comment about the overall
>> design. If it will turn out to "look good", then we'll select some
>> better names for new flag and data types.
>>
>> I use the last bit in the clone_flags for CLONE_LONGARG. When set it
>> will denote that the child_tidptr is not a pointer to a tid storage,
>> but the pointer to the struct long_clone_struct which currently
>> looks like this:
>>
>> struct long_clone_arg {
>>  int size;
>> };
>
> Ugly as night, I'd say. (Al said it better). What about just adding
> clone2 syscall, that takes u64?

yes but we would need more something like :

 long sys_clone64(unsigned long flags_high, unsigned long flag_low)

if we want the syscall to be supported on 32bit arch. clone2 is also
being used on ia64 already.

C.

_____

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Dave Hansen on Thu, 24 Jan 2008 17:24:34 GMT
View Forum Message <> Reply to Message

On Thu, 2008-01-24 at 18:09 +0100, Cedric Le Goater wrote:
> yes but we would need more something like :
>
> long sys_clone64(unsigned long flags_high, unsigned long flag_low)
>
> if we want the syscall to be supported on 32bit arch. clone2 is also
> being used on ia64 already.

Did we decide not to do something with a variable number of arguments?

sys_clonefoo(unsigned long *flags, int len);

-- Dave

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

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Pavel Machek on Thu, 24 Jan 2008 17:37:08 GMT
View Forum Message <> Reply to Message

On Thu 2008-01-24 09:24:34, Dave Hansen wrote:
> On Thu, 2008-01-24 at 18:09 +0100, Cedric Le Goater wrote:
> > yes but we would need more something like :
> >
> > long sys_clone64(unsigned long flags_high, unsigned long flag_low)
> >
> > if we want the syscall to be supported on 32bit arch. clone2 is also
> > being used on ia64 already.
>
> Did we decide not to do something with a variable number of arguments?
>
> sys_clonefoo(unsigned long *flags, int len);

That is evil, because that means strace can no longer reliably print

flags being used (for example).
	Pavel
--
(english) http://www.livejournal.com/~pavelmachek
(cesky, pictures) http://atrey.karlin.mff.cuni.cz/~pavel/picture/horses/blog.html

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

---

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Dave Hansen on Thu, 24 Jan 2008 17:46:57 GMT
View Forum Message <> Reply to Message

On Thu, 2008-01-24 at 18:37 +0100, Pavel Machek wrote:
> > Did we decide not to do something with a variable number of
> arguments?
> >
> > 	sys_clonefoo(unsigned long *flags, int len);
>
> That is evil, because that means strace can no longer reliably print
> flags being used (for example).

Ahhh. Just like it can't print strings for "buf"?

write(int fd, char *buf, size_t len)

Man, strace is stupid! ;)

If that's *really* a concern, why don't we just pass, say 4 or 5 longs
in:

sys_clonebig(unsigned long flags0, unsigned long flags1,
	unsigned long flags2, unsigned long flags3);

-- Dave

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

---

Subject: Re: [PATCH 1/2] Extend sys_clone and sys_unshare system calls API
Posted by Pavel Machek on Thu, 24 Jan 2008 17:50:33 GMT
View Forum Message <> Reply to Message

On Thu 2008-01-24 09:46:57, Dave Hansen wrote:
> On Thu, 2008-01-24 at 18:37 +0100, Pavel Machek wrote:
> > > Did we decide not to do something with a variable number of
> > arguments?
> > >
> > >  sys_clonefoo(unsigned long *flags, int len);
> > >
> > That is evil, because that means strace can no longer reliably print
> > flags being used (for example).
>
> Ahhh.  Just like it can't print strings for "buf"?
>
>  write(int fd, char *buf, size_t len)
>
> Man, strace is stupid! ;)

I said _reliably_.

Guess what happens on smp when one process will periodically overwrite
buf, and second will do the syscall?

It sucks for write, but it would suck much more for clone, where flags
drastically change the behaviour.

(Yep, it is possible to do reliably, see subterfugue, but it is a lot
of work and lot of overhead).

> If that's *really* a concern, why don't we just pass, say 4 or 5 longs
> in:
>
>  sys_clonebig(unsigned long flags0, unsigned long flags1,
>       unsigned long flags2, unsigned long flags3);

Works for me.
        Pavel
--
(english) http://www.livejournal.com/~pavelmachek
(cesky, pictures) http://atrey.karlin.mff.cuni.cz/~pavel/picture/horses/blog.html
_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers