Subject: [PATCH 0/5][V2] Misc helper patches for pid namespaces
Posted by Sukadev Bhattiprolu on Thu, 19 Jul 2007 07:12:06 GMT
View Forum Message <> Reply to Message

Some helper patches to support multiple pid namespaces. These
were posted earlier on Containers@ mailing list.

[PATCH 1/5] Define and use task_active_pid_ns() wrapper
[PATCH 2/5] Rename child_reaper() function.
[PATCH 3/5] Use task_pid() to find leader's pid
[PATCH 4/5] Define is_global_init() and is_container_init().
[PATCH 5/5] Move alloc_pid() to copy_process()

Changelog:

- Addressed Oleg Nesterov's comments on [V1] of this patchset.

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

Subject: [PATCH 1/5] [V2] Define and use task_active_pid_ns() wrapper
Posted by Sukadev Bhattiprolu on Thu, 19 Jul 2007 07:19:36 GMT
View Forum Message <> Reply to Message

Subject: [PATCH 1/5] Define and use task_active_pid_ns() wrapper

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

With multiple pid namespaces, a process is known by some pid_t in
every ancestor pid namespace. Every time the process forks, the
child process also gets a pid_t in every ancestor pid namespace.

While a process is visible in >=1 pid namespaces, it can see pid_t's
in only one pid namespace. We call this pid namespace it's "active
pid namespace", and it is always the youngest pid namespace in which
the process is known.

This patch defines and uses a wrapper to find the active pid namespace
of a process. The implementation of the wrapper will be changed in
when support for multiple pid namespaces are added.

Changelog:
2.6.22-rc4-mm2-pidns1:
- [Pavel Emelianov, Alexey Dobriyan] Back out the change to use
task_active_pid_ns() in child_reaper() since task->nsproxy
can be NULL during task exit (so child_reaper() continues to

```
   use init_pid_ns).

   to implement child_reaper() since init_pid_ns.child_reaper to
   implement child_reaper() since tsk->nsproxy can be NULL during exit.

 2.6.21-rc6-mm1:
 - Rename task_pid_ns() to task_active_pid_ns() to reflect that a
   process can have multiple pid namespaces.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Acked-by: Pavel Emelianov <xemul@openvz.org>

Cc: Eric W. Biederman <ebiederm@xmission.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: Herbert Poetzel <herbert@13thfloor.at>
---
 fs/exec.c                  |    2 +-
 fs/proc/proc_misc.c        |    3 ++-
 include/linux/pid_namespace.h |    7 ++++++-
 kernel/exit.c              |    5 +++--
 kernel/nsproxy.c           |    2 +-
 kernel/pid.c               |    4 ++--
 6 files changed, 15 insertions(+), 8 deletions(-)

Index: lx26-22-rc6-mm1/include/linux/pid_namespace.h
===================================================================
--- lx26-22-rc6-mm1.orig/include/linux/pid_namespace.h 2007-07-13 13:07:01.000000000 -0700
+++ lx26-22-rc6-mm1/include/linux/pid_namespace.h 2007-07-13 18:22:49.000000000 -0700
@@ -20,7 +20,7 @@ struct pid_namespace {
  struct pidmap pidmap[PIDMAP_ENTRIES];
  int last_pid;
  struct task_struct *child_reaper;
- struct kmem_cache_t *pid_cachep;
+ struct kmem_cache *pid_cachep;
 };

 extern struct pid_namespace init_pid_ns;
@@ -39,6 +39,11 @@ static inline void put_pid_ns(struct pid
  kref_put(&ns->kref, free_pid_ns);
 }

+static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
+{
+ return tsk->nsproxy->pid_ns;
+}
+
```

```
 static inline struct task_struct *child_reaper(struct task_struct *tsk)
 {
  return init_pid_ns.child_reaper;
Index: lx26-22-rc6-mm1/fs/exec.c
===================================================================
--- lx26-22-rc6-mm1.orig/fs/exec.c 2007-07-13 13:05:38.000000000 -0700
+++ lx26-22-rc6-mm1/fs/exec.c 2007-07-13 18:13:39.000000000 -0700
@@ -827,7 +827,7 @@ static int de_thread(struct task_struct
   * so it is safe to do it under read_lock.
   */
  if (unlikely(tsk->group_leader == child_reaper(tsk)))
-  tsk->nsproxy->pid_ns->child_reaper = tsk;
+  task_active_pid_ns(tsk)->child_reaper = tsk;

  zap_other_threads(tsk);
  read_unlock(&tasklist_lock);
Index: lx26-22-rc6-mm1/fs/proc/proc_misc.c
===================================================================
--- lx26-22-rc6-mm1.orig/fs/proc/proc_misc.c 2007-07-13 13:05:38.000000000 -0700
+++ lx26-22-rc6-mm1/fs/proc/proc_misc.c 2007-07-13 13:07:48.000000000 -0700
@@ -94,7 +94,8 @@ static int loadavg_read_proc(char *page,
   LOAD_INT(a), LOAD_FRAC(a),
   LOAD_INT(b), LOAD_FRAC(b),
   LOAD_INT(c), LOAD_FRAC(c),
-  nr_running(), nr_threads, current->nsproxy->pid_ns->last_pid);
+  nr_running(), nr_threads,
+  task_active_pid_ns(current)->last_pid);
  return proc_calc_metrics(page, start, off, count, eof, len);
 }

Index: lx26-22-rc6-mm1/kernel/exit.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/exit.c 2007-07-13 13:06:52.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/exit.c 2007-07-13 18:13:39.000000000 -0700
@@ -909,8 +909,9 @@ fastcall NORET_TYPE void do_exit(long co
  if (unlikely(!tsk->pid))
   panic("Attempted to kill the idle task!");
  if (unlikely(tsk == child_reaper(tsk))) {
-  if (tsk->nsproxy->pid_ns != &init_pid_ns)
-   tsk->nsproxy->pid_ns->child_reaper = init_pid_ns.child_reaper;
+  if (task_active_pid_ns(tsk) != &init_pid_ns)
+   task_active_pid_ns(tsk)->child_reaper =
+     init_pid_ns.child_reaper;
   else
    panic("Attempted to kill init!");
  }
Index: lx26-22-rc6-mm1/kernel/pid.c
===================================================================
```

```
--- lx26-22-rc6-mm1.orig/kernel/pid.c 2007-07-13 13:07:01.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/pid.c 2007-07-13 18:13:38.000000000 -0700
@@ -214,7 +214,7 @@ struct pid *alloc_pid(void)
  int nr = -1;
  struct pid_namespace *ns;

- ns = current->nsproxy->pid_ns;
+ ns = task_active_pid_ns(current);
  pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
  if (!pid)
   goto out;
@@ -364,7 +364,7 @@ struct pid *find_ge_pid(int nr)
   pid = find_pid(nr);
   if (pid)
    break;
-  nr = next_pidmap(current->nsproxy->pid_ns, nr);
+  nr = next_pidmap(task_active_pid_ns(current), nr);
  } while (nr > 0);

  return pid;
Index: lx26-22-rc6-mm1/kernel/nsproxy.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/nsproxy.c 2007-07-13 13:05:38.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/nsproxy.c 2007-07-13 13:07:48.000000000 -0700
@@ -86,7 +86,7 @@ static struct nsproxy *create_new_namesp
   goto out_ipc;
  }

- new_nsp->pid_ns = copy_pid_ns(flags, tsk->nsproxy->pid_ns);
+ new_nsp->pid_ns = copy_pid_ns(flags, task_active_pid_ns(tsk));
  if (IS_ERR(new_nsp->pid_ns)) {
   err = PTR_ERR(new_nsp->pid_ns);
   goto out_pid;
```

_____

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

---

Subject: [PATCH 2/5] [V2] Rename child_reaper() function
Posted by Sukadev Bhattiprolu on Thu, 19 Jul 2007 07:20:11 GMT
View Forum Message <> Reply to Message

Pavel,

Pls ack this if you agree.

Suka

Subject: [PATCH 2/5] Rename child_reaper function.

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Rename the child_reaper() function to task_child_reaper() to be
similar to other task_* functions and to distinguish the function
from 'struct pid_namspace.child_reaper'.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
 fs/exec.c | 2 +-
 include/linux/pid_namespace.h | 2 +-
 kernel/exit.c | 4 ++--
 kernel/signal.c | 2 +-
 4 files changed, 5 insertions(+), 5 deletions(-)

Index: lx26-22-rc6-mm1/include/linux/pid_namespace.h
==========
--- lx26-22-rc6-mm1.orig/include/linux/pid_namespace.h 2007-07-13 13:07:48.000000000 -0700
+++ lx26-22-rc6-mm1/include/linux/pid_namespace.h 2007-07-13 13:12:01.000000000 -0700
@@ -44,7 +44,7 @@ static inline struct pid_namespace *task
  return tsk->nsproxy->pid_ns;
 }

-static inline struct task_struct *child_reaper(struct task_struct *tsk)
+static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
 {
  return init_pid_ns.child_reaper;
 }
Index: lx26-22-rc6-mm1/fs/exec.c
==========
--- lx26-22-rc6-mm1.orig/fs/exec.c 2007-07-13 13:07:48.000000000 -0700
+++ lx26-22-rc6-mm1/fs/exec.c 2007-07-13 13:12:01.000000000 -0700
@@ -826,7 +826,7 @@ static int de_thread(struct task_struct
   * Reparenting needs write_lock on tasklist_lock,
   * so it is safe to do it under read_lock.
   */
- if (unlikely(tsk->group_leader == child_reaper(tsk)))
+ if (unlikely(tsk->group_leader == task_child_reaper(tsk)))
   task_active_pid_ns(tsk)->child_reaper = tsk;

  zap_other_threads(tsk);
Index: lx26-22-rc6-mm1/kernel/exit.c
==========
--- lx26-22-rc6-mm1.orig/kernel/exit.c 2007-07-13 13:07:48.000000000 -0700

```
+++ lx26-22-rc6-mm1/kernel/exit.c 2007-07-13 13:12:01.000000000 -0700
@@ -695,7 +695,7 @@ forget_original_parent(struct task_struc
  do {
   reaper = next_thread(reaper);
   if (reaper == father) {
-   reaper = child_reaper(father);
+   reaper = task_child_reaper(father);
    break;
   }
  } while (reaper->exit_state);
@@ -908,7 +908,7 @@ fastcall NORET_TYPE void do_exit(long co
   panic("Aiee, killing interrupt handler!");
  if (unlikely(!tsk->pid))
   panic("Attempted to kill the idle task!");
- if (unlikely(tsk == child_reaper(tsk))) {
+ if (unlikely(tsk == task_child_reaper(tsk))) {
   if (task_active_pid_ns(tsk) != &init_pid_ns)
    task_active_pid_ns(tsk)->child_reaper =
      init_pid_ns.child_reaper;
Index: lx26-22-rc6-mm1/kernel/signal.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/signal.c 2007-07-13 13:06:52.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/signal.c 2007-07-13 13:12:01.000000000 -0700
@@ -1853,7 +1853,7 @@ relock:
    * within that pid space. It can of course get signals from
    * its parent pid space.
    */
-  if (current == child_reaper(current))
+  if (current == task_child_reaper(current))
    continue;

   if (sig_kernel_stop(signr)) {
```

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

---

Subject: [PATCH 3/5] [V2] Use task_pid() to find leader's pid
Posted by Sukadev Bhattiprolu on Thu, 19 Jul 2007 07:20:52 GMT
View Forum Message <> Reply to Message

Subject: [PATCH 3/5] Use task_pid() to find leader's pid

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Use task_pid() to get leader's 'struct pid' and avoid the find_pid().

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Acked-by: Pavel Emelianov <xemul@openvz.org>

Cc: Eric W. Biederman <ebiederm@xmission.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: Herbert Poetzel <herbert@13thfloor.at>
---
 fs/exec.c | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Index: lx26-22-rc6-mm1a/fs/exec.c
==========
--- lx26-22-rc6-mm1a.orig/fs/exec.c 2007-07-13 18:23:55.000000000 -0700
+++ lx26-22-rc6-mm1a/fs/exec.c 2007-07-16 12:56:22.000000000 -0700
@@ -908,7 +908,7 @@ static int de_thread(struct task_struct
    */
   detach_pid(tsk, PIDTYPE_PID);
   tsk->pid = leader->pid;
- attach_pid(tsk, PIDTYPE_PID, find_pid(tsk->pid));
+ attach_pid(tsk, PIDTYPE_PID, task_pid(leader));
   transfer_pid(leader, tsk, PIDTYPE_PGID);
   transfer_pid(leader, tsk, PIDTYPE_SID);
   list_replace_rcu(&leader->tasks, &tsk->tasks);

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

---

Subject: [PATCH 4/5] [V2] Define is_global_init() and is_container_init()
Posted by Sukadev Bhattiprolu on Thu, 19 Jul 2007 07:21:58 GMT
View Forum Message <> Reply to Message

Subject: [PATCH 4/5] Define is_global_init() and is_container_init().

From: Serge E. Hallyn <serue@us.ibm.com>

is_init() is an ambiguous name for the pid==1 check. Split it into
is_global_init() and is_container_init().

A container init has it's tsk->pid == 1.

A global init also has it's tsk->pid == 1 and it's active pid namespace
is the init_pid_ns. But rather than check the active pid namespace,
compare the task structure with 'init_pid_ns.child_reaper', which is

initialized during boot to the /sbin/init process and never changes.

Changelog:

2.6.22-rc4-mm2-pidns1:
- Use 'init_pid_ns.child_reaper' to determine if a given task is the
  global init (/sbin/init) process. This would improve performance
  and remove dependence on the task_pid().

2.6.21-mm2-pidns2:

- [Sukadev Bhattiprolu] Changed is_container_init() calls in {powerpc,
  ppc,avr32}/traps.c for the _exception() call to is_global_init().
  This way, we kill only the container if the container's init has a
  bug rather than force a kernel panic.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Acked-by: Pavel Emelianov <xemul@openvz.org>

Cc: Eric W. Biederman <ebiederm@xmission.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Herbert Poetzel <herbert@13thfloor.at>
---
arch/alpha/mm/fault.c | 2 +-
arch/arm/mm/fault.c | 2 +-
arch/arm26/mm/fault.c | 2 +-
arch/avr32/kernel/traps.c | 2 +-
arch/avr32/mm/fault.c | 6 +++---
arch/i386/lib/usercopy.c | 2 +-
arch/i386/mm/fault.c | 2 +-
arch/ia64/mm/fault.c | 2 +-
arch/m68k/mm/fault.c | 2 +-
arch/mips/mm/fault.c | 2 +-
arch/powerpc/kernel/traps.c | 2 +-
arch/powerpc/mm/fault.c | 2 +-
arch/powerpc/platforms/pseries/ras.c | 2 +-
arch/ppc/kernel/traps.c | 2 +-
arch/ppc/mm/fault.c | 2 +-
arch/s390/lib/uaccess_pt.c | 2 +-
arch/s390/mm/fault.c | 2 +-
arch/sh/mm/fault.c | 2 +-
arch/sh64/mm/fault.c | 6 +++---
arch/um/kernel/trap.c | 2 +-
arch/x86_64/mm/fault.c | 2 +-
arch/xtensa/mm/fault.c | 2 +-
drivers/char/sysrq.c | 2 +-

```
 include/linux/sched.h              |   12 ++++++++++--
 kernel/capability.c                |    3 ++-
 kernel/exit.c                      |    2 +-
 kernel/kexec.c                     |    2 +-
 kernel/pid.c                       |    7 +++++++
 kernel/signal.c                    |    2 +-
 kernel/sysctl.c                    |    2 +-
 mm/oom_kill.c                      |    4 ++--
 security/commoncap.c               |    3 ++-
 32 files changed, 54 insertions(+), 37 deletions(-)

Index: lx26-22-rc6-mm1a/include/linux/sched.h
===================================================================
--- lx26-22-rc6-mm1a.orig/include/linux/sched.h 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/include/linux/sched.h 2007-07-16 13:10:48.000000000 -0700
@@ -1219,12 +1219,20 @@ static inline int pid_alive(struct task_
 }

 /**
- * is_init - check if a task structure is init
+ * is_global_init - check if a task structure is init
  * @tsk: Task structure to be checked.
  *
  * Check if a task structure is the first user space task the kernel created.
+ *
+ * TODO: We should inline this function after some cleanups in pid_namespace.h
+ */
+extern int is_global_init(struct task_struct *tsk);
+
+/*
+ * is_container_init:
+ * check whether in the task is init in it's own pid namespace.
  */
-static inline int is_init(struct task_struct *tsk)
+static inline int is_container_init(struct task_struct *tsk)
 {
  return tsk->pid == 1;
 }
Index: lx26-22-rc6-mm1a/kernel/pid.c
===================================================================
--- lx26-22-rc6-mm1a.orig/kernel/pid.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/pid.c 2007-07-16 13:10:48.000000000 -0700
@@ -69,6 +69,13 @@ struct pid_namespace init_pid_ns = {
  .last_pid = 0,
  .child_reaper = &init_task
 };
+EXPORT_SYMBOL(init_pid_ns);
+
```

```
+int is_global_init(struct task_struct *tsk)
+{
+ return tsk == init_pid_ns.child_reaper;
+}
+EXPORT_SYMBOL(is_global_init);

 /*
  * Note: disable interrupts while the pidmap_lock is held as an
Index: lx26-22-rc6-mm1a/arch/alpha/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/alpha/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/alpha/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -192,7 +192,7 @@ do_page_fault(unsigned long address, uns
  /* We ran out of memory, or some other thing happened to us that
     made us unable to handle the page fault gracefully.  */
  out_of_memory:
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/arm/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/arm/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/arm/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -197,7 +197,7 @@ survive:
   return fault;
  }

- if (!is_init(tsk))
+ if (!is_global_init(tsk))
   goto out;

  /*
Index: lx26-22-rc6-mm1a/arch/arm26/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/arm26/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/arm26/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -185,7 +185,7 @@ survive:
  }

  fault = -3; /* out of memory */
- if (!is_init(tsk))
+ if (!is_global_init(tsk))
   goto out;

  /*
Index: lx26-22-rc6-mm1a/arch/i386/lib/usercopy.c
```

```
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/i386/lib/usercopy.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/i386/lib/usercopy.c 2007-07-16 13:10:48.000000000 -0700
@@ -748,7 +748,7 @@ survive:
    retval = get_user_pages(current, current->mm,
      (unsigned long )to, 1, 1, 0, &pg, NULL);

-   if (retval == -ENOMEM && is_init(current)) {
+   if (retval == -ENOMEM && is_global_init(current)) {
     up_read(&current->mm->mmap_sem);
     congestion_wait(WRITE, HZ/50);
     goto survive;
Index: lx26-22-rc6-mm1a/arch/i386/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/i386/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/i386/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -595,7 +595,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(tsk)) {
+ if (is_global_init(tsk)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/ia64/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/ia64/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/ia64/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -270,7 +270,7 @@ ia64_do_page_fault (unsigned long addres

   out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/m68k/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/m68k/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/m68k/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -181,7 +181,7 @@ good_area:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
```

```
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/mips/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/mips/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/mips/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -174,7 +174,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(tsk)) {
+ if (is_global_init(tsk)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/powerpc/kernel/traps.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/powerpc/kernel/traps.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/powerpc/kernel/traps.c 2007-07-16 13:10:48.000000000 -0700
@@ -191,7 +191,7 @@ void _exception(int signr, struct pt_reg
   * generate the same exception over and over again and we get
   * nowhere.  Better to kill it and let the kernel panic.
   */
- if (is_init(current)) {
+ if (is_global_init(current)) {
   __sighandler_t handler;

   spin_lock_irq(&current->sighand->siglock);
Index: lx26-22-rc6-mm1a/arch/powerpc/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/powerpc/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/powerpc/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -374,7 +374,7 @@ bad_area_nosemaphore:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/powerpc/platforms/pseries/ras.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/powerpc/platforms/pseries/ras.c 2007-07-16 12:55:15.000000000
-0700
+++ lx26-22-rc6-mm1a/arch/powerpc/platforms/pseries/ras.c 2007-07-16 13:10:48.000000000
-0700
```

```
@@ -332,7 +332,7 @@ static int recover_mce(struct pt_regs *r
      err->disposition == RTAS_DISP_NOT_RECOVERED &&
      err->target == RTAS_TARGET_MEMORY &&
      err->type == RTAS_TYPE_ECC_UNCORR &&
-     !(current->pid == 0 || is_init(current))) {
+     !(current->pid == 0 || is_global_init(current))) {
   /* Kill off a user process with an ECC error */
   printk(KERN_ERR "MCE: uncorrectable ecc error for pid %d\n",
          current->pid);
Index: lx26-22-rc6-mm1a/arch/ppc/kernel/traps.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/ppc/kernel/traps.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/ppc/kernel/traps.c 2007-07-16 13:10:48.000000000 -0700
@@ -121,7 +121,7 @@ void _exception(int signr, struct pt_reg
   * generate the same exception over and over again and we get
   * nowhere.  Better to kill it and let the kernel panic.
   */
- if (is_init(current)) {
+ if (is_global_init(current)) {
   __sighandler_t handler;

   spin_lock_irq(&current->sighand->siglock);
Index: lx26-22-rc6-mm1a/arch/ppc/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/ppc/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/ppc/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -292,7 +292,7 @@ bad_area:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/s390/lib/uaccess_pt.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/s390/lib/uaccess_pt.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/s390/lib/uaccess_pt.c 2007-07-16 13:10:48.000000000 -0700
@@ -65,7 +65,7 @@ out:

 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
```

```
Index: lx26-22-rc6-mm1a/arch/s390/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/s390/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/s390/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -211,7 +211,7 @@ static int do_out_of_memory(struct pt_re
  struct mm_struct *mm = tsk->mm;

  up_read(&mm->mmap_sem);
- if (is_init(tsk)) {
+ if (is_global_init(tsk)) {
   yield();
   down_read(&mm->mmap_sem);
   return 1;
Index: lx26-22-rc6-mm1a/arch/sh/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/sh/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/sh/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -191,7 +191,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/sh64/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/sh64/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/sh64/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -276,7 +276,7 @@ bad_area:
    show_regs(regs);
 #endif
   }
-  if (is_init(tsk)) {
+  if (is_global_init(tsk)) {
    panic("INIT had user mode bad_area\n");
   }
   tsk->thread.address = address;
@@ -318,14 +318,14 @@ no_context:
  * us unable to handle the page fault gracefully.
  */
 out_of_memory:
- if (is_init(current)) {
+ if (is_global_init(current)) {
   panic("INIT out of memory\n");
   yield();
   goto survive;
```

```
  }
  printk("fault:Out of memory\n");
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/arch/um/kernel/trap.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/um/kernel/trap.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/um/kernel/trap.c 2007-07-16 13:10:48.000000000 -0700
@@ -120,7 +120,7 @@ out_nosemaphore:
  * us unable to handle the page fault gracefully.
  */
 out_of_memory:
- if (is_init(current)) {
+ if (is_global_init(current)) {
   up_read(&mm->mmap_sem);
   yield();
   down_read(&mm->mmap_sem);
Index: lx26-22-rc6-mm1a/arch/x86_64/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/x86_64/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/x86_64/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -558,7 +558,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   goto again;
  }
Index: lx26-22-rc6-mm1a/arch/xtensa/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/xtensa/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/xtensa/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -144,7 +144,7 @@ bad_area:
   */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1a/drivers/char/sysrq.c
```

```
===================================================================
--- lx26-22-rc6-mm1a.orig/drivers/char/sysrq.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/drivers/char/sysrq.c 2007-07-16 13:10:48.000000000 -0700
@@ -250,7 +250,7 @@ static void send_sig_all(int sig)
  struct task_struct *p;

  for_each_process(p) {
-  if (p->mm && !is_init(p))
+  if (p->mm && !is_global_init(p))
    /* Not swapper, init nor kernel thread */
    force_sig(sig, p);
  }
Index: lx26-22-rc6-mm1a/kernel/capability.c
===================================================================
--- lx26-22-rc6-mm1a.orig/kernel/capability.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/capability.c 2007-07-16 13:10:48.000000000 -0700
@@ -12,6 +12,7 @@
 #include <linux/module.h>
 #include <linux/security.h>
 #include <linux/syscalls.h>
+#include <linux/pid_namespace.h>
 #include <asm/uaccess.h>

 unsigned securebits = SECUREBITS_DEFAULT; /* systemwide security settings */
@@ -135,7 +136,7 @@ static inline int cap_set_all(kernel_cap
      int found = 0;

      do_each_thread(g, target) {
-             if (target == current || is_init(target))
+             if (target == current || is_container_init(target))
                     continue;
             found = 1;
       if (security_capset_check(target, effective, inheritable,
Index: lx26-22-rc6-mm1a/kernel/exit.c
===================================================================
--- lx26-22-rc6-mm1a.orig/kernel/exit.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/exit.c 2007-07-16 13:10:48.000000000 -0700
@@ -231,7 +231,7 @@ static int will_become_orphaned_pgrp(str
  do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
   if (p == ignored_task
     || p->exit_state
-    || is_init(p->real_parent))
+    || is_global_init(p->real_parent))
    continue;
   if (task_pgrp(p->real_parent) != pgrp &&
       task_session(p->real_parent) == task_session(p)) {
Index: lx26-22-rc6-mm1a/kernel/kexec.c
===================================================================
```

```
--- lx26-22-rc6-mm1a.orig/kernel/kexec.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/kexec.c 2007-07-16 13:10:48.000000000 -0700
@@ -42,7 +42,7 @@ struct resource crashk_res = {

 int kexec_should_crash(struct task_struct *p)
 {
- if (in_interrupt() || !p->pid || is_init(p) || panic_on_oops)
+ if (in_interrupt() || !p->pid || is_global_init(p) || panic_on_oops)
    return 1;
  return 0;
 }
Index: lx26-22-rc6-mm1a/kernel/sysctl.c
===================================================================
--- lx26-22-rc6-mm1a.orig/kernel/sysctl.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/sysctl.c 2007-07-16 13:10:48.000000000 -0700
@@ -1924,7 +1924,7 @@ int proc_dointvec_bset(ctl_table *table,
    return -EPERM;
  }

- op = is_init(current) ? OP_SET : OP_AND;
+ op = is_global_init(current) ? OP_SET : OP_AND;
  return do_proc_dointvec(table,write,filp,buffer,lenp,ppos,
     do_proc_dointvec_bset_conv,&op);
 }
Index: lx26-22-rc6-mm1a/mm/oom_kill.c
===================================================================
--- lx26-22-rc6-mm1a.orig/mm/oom_kill.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/mm/oom_kill.c 2007-07-16 13:10:48.000000000 -0700
@@ -222,7 +222,7 @@ static struct task_struct *select_bad_pr
    if (!p->mm)
     continue;
    /* skip the init task */
-  if (is_init(p))
+  if (is_global_init(p))
     continue;

    /*
@@ -275,7 +275,7 @@ static struct task_struct *select_bad_pr
  */
 static void __oom_kill_task(struct task_struct *p, int verbose)
 {
- if (is_init(p)) {
+ if (is_global_init(p)) {
    WARN_ON(1);
    printk(KERN_WARNING "tried to kill init!\n");
    return;
Index: lx26-22-rc6-mm1a/security/commoncap.c
===================================================================
```

```
--- lx26-22-rc6-mm1a.orig/security/commoncap.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/security/commoncap.c 2007-07-16 13:10:48.000000000 -0700
@@ -23,6 +23,7 @@
 #include <linux/xattr.h>
 #include <linux/hugetlb.h>
 #include <linux/mount.h>
+#include <linux/sched.h>

 int cap_netlink_send(struct sock *sk, struct sk_buff *skb)
 {
@@ -261,7 +262,7 @@ void cap_bprm_apply_creds (struct linux_
  /* For init, we want to retain the capabilities set
   * in the init_task struct. Thus we skip the usual
   * capability rules */
- if (!is_init(current)) {
+ if (!is_global_init(current)) {
   current->cap_permitted = new_permitted;
   current->cap_effective =
       cap_intersect (new_permitted, bprm->cap_effective);
Index: lx26-22-rc6-mm1a/arch/avr32/kernel/traps.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/avr32/kernel/traps.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/avr32/kernel/traps.c 2007-07-16 13:10:48.000000000 -0700
@@ -89,7 +89,7 @@ void _exception(long signr, struct pt_re
   * generate the same exception over and over again and we get
   * nowhere.  Better to kill it and let the kernel panic.
   */
- if (is_init(current)) {
+ if (is_global_init(current)) {
   __sighandler_t handler;

   spin_lock_irq(&current->sighand->siglock);
Index: lx26-22-rc6-mm1a/arch/avr32/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1a.orig/arch/avr32/mm/fault.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/arch/avr32/mm/fault.c 2007-07-16 13:10:48.000000000 -0700
@@ -161,7 +161,7 @@ bad_area:
   if (exception_trace && printk_ratelimit())
    printk("%s%s[%d]: segfault at %08lx pc %08lx "
           "sp %08lx ecr %lu\n",
-          is_init(tsk) ? KERN_EMERG : KERN_INFO,
+          is_global_init(tsk) ? KERN_EMERG : KERN_INFO,
           tsk->comm, tsk->pid, address, regs->pc,
           regs->sp, ecr);
   _exception(SIGSEGV, regs, code, address);
@@ -210,7 +210,7 @@ no_context:
   */
 out_of_memory:
```

```
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
@@ -232,7 +232,7 @@ do_sigbus:
  if (exception_trace)
   printk("%s%s[%d]: bus error at %08lx pc %08lx "
          "sp %08lx ecr %lu\n",
-         is_init(tsk) ? KERN_EMERG : KERN_INFO,
+         is_global_init(tsk) ? KERN_EMERG : KERN_INFO,
          tsk->comm, tsk->pid, address, regs->pc,
          regs->sp, ecr);

Index: lx26-22-rc6-mm1a/kernel/signal.c
===================================================================
--- lx26-22-rc6-mm1a.orig/kernel/signal.c 2007-07-16 12:55:15.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/signal.c 2007-07-16 13:10:48.000000000 -0700
@@ -257,7 +257,7 @@ flush_signal_handlers(struct task_struct

 int unhandled_signal(struct task_struct *tsk, int sig)
 {
- if (is_init(tsk))
+ if (is_global_init(tsk))
   return 1;
  if (tsk->ptrace & PT_PTRACED)
   return 0;
```

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

---

Subject: [PATCH 5/5] [V2] Move alloc_pid() to copy_process()
Posted by Sukadev Bhattiprolu on Thu, 19 Jul 2007 07:22:40 GMT
View Forum Message <> Reply to Message

Subject: [PATCH 5/5] Move alloc_pid call to copy_process

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Move alloc_pid() into copy_process(). This will keep all pid and pid namespace code together and simplify error handling when we support multiple pid namespaces.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

```
Cc: Pavel Emelianov <xemul@openvz.org>
Cc: Eric W. Biederman <ebiederm@xmission.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: Herbert Poetzel <herbert@13thfloor.at>
---
 kernel/fork.c |   19 ++++++++++++------
 1 file changed, 13 insertions(+), 6 deletions(-)

Index: lx26-22-rc6-mm1a/kernel/fork.c
===================================================================
--- lx26-22-rc6-mm1a.orig/kernel/fork.c 2007-07-16 12:55:13.000000000 -0700
+++ lx26-22-rc6-mm1a/kernel/fork.c 2007-07-17 10:08:12.000000000 -0700
@@ -1029,6 +1029,12 @@ static struct task_struct *copy_process(
  if (p->binfmt && !try_module_get(p->binfmt->module))
   goto bad_fork_cleanup_put_domain;

+ if (pid != &init_struct_pid) {
+  pid = alloc_pid();
+  if (!pid)
+   goto bad_fork_put_binfmt_module;
+ }
+
  p->did_exec = 0;
  delayacct_tsk_init(p); /* Must remain after dup_task_struct() */
  copy_flags(clone_flags, p);
@@ -1316,6 +1322,9 @@ bad_fork_cleanup_container:
 #endif
  container_exit(p, container_callbacks_done);
  delayacct_tsk_free(p);
+ if (pid != &init_struct_pid)
+  free_pid(pid);
+bad_fork_put_binfmt_module:
  if (p->binfmt)
   module_put(p->binfmt->module);
 bad_fork_cleanup_put_domain:
@@ -1380,19 +1389,16 @@ long do_fork(unsigned long clone_flags,
 {
  struct task_struct *p;
  int trace = 0;
- struct pid *pid = alloc_pid();
  long nr;

- if (!pid)
-  return -EAGAIN;
- nr = pid->nr;
  if (unlikely(current->ptrace)) {
```

```
   trace = fork_traceflag (clone_flags);
   if (trace)
    clone_flags |= CLONE_PTRACE;
  }

- p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr, child_tidptr, pid);
+ p = copy_process(clone_flags, stack_start, regs, stack_size,
+    parent_tidptr, child_tidptr, NULL);
   /*
    * Do this prior waking up the new thread - the thread pointer
    * might get invalid after that point, if the thread exits quickly.
@@ -1400,6 +1406,8 @@ long do_fork(unsigned long clone_flags,
  if (!IS_ERR(p)) {
   struct completion vfork;

+  nr = pid_nr(task_pid(p));
+
   if (clone_flags & CLONE_VFORK) {
    p->vfork_done = &vfork;
    init_completion(&vfork);
@@ -1433,7 +1441,6 @@ long do_fork(unsigned long clone_flags,
    }
   }
  } else {
-  free_pid(pid);
   nr = PTR_ERR(p);
  }
  return nr;
```

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

---

Subject: Re: [PATCH 5/5] [V2] Move alloc_pid() to copy_process()
Posted by Pavel Emelianov on Thu, 19 Jul 2007 07:44:47 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
>
> Subject: [PATCH 5/5] Move alloc_pid call to copy_process
>
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>
> Move alloc_pid() into copy_process(). This will keep all pid and pid
> namespace code together and simplify error handling when we support
> multiple pid namespaces.

I would add smth like this to the comment:

When a task creates a new pid namespace, its init (i.e. this task's
child) will have pids with extra info inside - the new numerical id,
that represent this new task in this new namespace. Thus, we have
to allocate this new pid only after the namespace creation to find
out which namespace this pid will live in.

Hope, I expressed my idea cleanly.

Acked-by: Pavel Emelyanov <xemul@openvz.org>

> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>
> Cc: Pavel Emelianov <xemul@openvz.org>
> Cc: Eric W. Biederman <ebiederm@xmission.com>
> Cc: Cedric Le Goater <clg@fr.ibm.com>
> Cc: Dave Hansen <haveblue@us.ibm.com>
> Cc: Serge Hallyn <serue@us.ibm.com>
> Cc: Herbert Poetzel <herbert@13thfloor.at>
> ---
> kernel/fork.c | 19 +++++++++++++------
> 1 file changed, 13 insertions(+), 6 deletions(-)
>
> Index: lx26-22-rc6-mm1a/kernel/fork.c
> ===
> --- lx26-22-rc6-mm1a.orig/kernel/fork.c 2007-07-16 12:55:13.000000000 -0700
> +++ lx26-22-rc6-mm1a/kernel/fork.c 2007-07-17 10:08:12.000000000 -0700
> @@ -1029,6 +1029,12 @@ static struct task_struct *copy_process(
>   if (p->binfmt && !try_module_get(p->binfmt->module))
>   goto bad_fork_cleanup_put_domain;
>
> + if (pid != &init_struct_pid) {
> + pid = alloc_pid();
> + if (!pid)
> +   goto bad_fork_put_binfmt_module;
> + }
> +
>   p->did_exec = 0;
>   delayacct_tsk_init(p); /* Must remain after dup_task_struct() */
>   copy_flags(clone_flags, p);
> @@ -1316,6 +1322,9 @@ bad_fork_cleanup_container:
> #endif
>   container_exit(p, container_callbacks_done);
>   delayacct_tsk_free(p);
> + if (pid != &init_struct_pid)
> + free_pid(pid);
> +bad_fork_put_binfmt_module:

```
>   if (p->binfmt)
>    module_put(p->binfmt->module);
>  bad_fork_cleanup_put_domain:
> @@ -1380,19 +1389,16 @@ long do_fork(unsigned long clone_flags,
>  {
>   struct task_struct *p;
>   int trace = 0;
> - struct pid *pid = alloc_pid();
>   long nr;
>
> - if (!pid)
> -  return -EAGAIN;
> - nr = pid->nr;
>   if (unlikely(current->ptrace)) {
>    trace = fork_traceflag (clone_flags);
>    if (trace)
>     clone_flags |= CLONE_PTRACE;
>   }
>
> - p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr, child_tidptr, pid);
> + p = copy_process(clone_flags, stack_start, regs, stack_size,
> +    parent_tidptr, child_tidptr, NULL);
>   /*
>    * Do this prior waking up the new thread - the thread pointer
>    * might get invalid after that point, if the thread exits quickly.
> @@ -1400,6 +1406,8 @@ long do_fork(unsigned long clone_flags,
>   if (!IS_ERR(p)) {
>    struct completion vfork;
>
> +  nr = pid_nr(task_pid(p));
> +
>    if (clone_flags & CLONE_VFORK) {
>     p->vfork_done = &vfork;
>     init_completion(&vfork);
> @@ -1433,7 +1441,6 @@ long do_fork(unsigned long clone_flags,
>     }
>    }
>   } else {
> -  free_pid(pid);
>    nr = PTR_ERR(p);
>   }
>   return nr;
>
```

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

Subject: Re: [PATCH 4/5] [V2] Define is_global_init() and is_container_init()
Posted by akpm on Fri, 20 Jul 2007 22:41:13 GMT
View Forum Message <> Reply to Message

On Thu, 19 Jul 2007 00:21:58 -0700
sukadev@us.ibm.com wrote:

> --- lx26-22-rc6-mm1a.orig/kernel/pid.c 2007-07-16 12:55:15.000000000 -0700
> +++ lx26-22-rc6-mm1a/kernel/pid.c 2007-07-16 13:10:48.000000000 -0700
> @@ -69,6 +69,13 @@ struct pid_namespace init_pid_ns = {
> .last_pid = 0,
> .child_reaper = &init_task
> };
> +EXPORT_SYMBOL(init_pid_ns);
> +
> +int is_global_init(struct task_struct *tsk)
> +{
> + return tsk == init_pid_ns.child_reaper;
> +}
> +EXPORT_SYMBOL(is_global_init);

I don't immediately see why init_pid_ns was exported to modules.

It would need to be exported if is_global_init() was made static inline in a
header (which seems like a sensible thing to do), but it wasn't.

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

Subject: Re: [PATCH 4/5] [V2] Define is_global_init() and is_container_init()
Posted by Sukadev Bhattiprolu on Sat, 21 Jul 2007 03:02:54 GMT
View Forum Message <> Reply to Message

Andrew Morton [akpm@linux-foundation.org] wrote:
| On Thu, 19 Jul 2007 00:21:58 -0700
| sukadev@us.ibm.com wrote:
|
| > --- lx26-22-rc6-mm1a.orig/kernel/pid.c 2007-07-16 12:55:15.000000000 -0700
| > +++ lx26-22-rc6-mm1a/kernel/pid.c 2007-07-16 13:10:48.000000000 -0700
| > @@ -69,6 +69,13 @@ struct pid_namespace init_pid_ns = {
| > .last_pid = 0,
| > .child_reaper = &init_task
| > };
| > +EXPORT_SYMBOL(init_pid_ns);
| > +

| > +int is_global_init(struct task_struct *tsk)
| > +{
| > + return tsk == init_pid_ns.child_reaper;
| > +}
| > +EXPORT_SYMBOL(is_global_init);
|
| I don't immediately see why init_pid_ns was exported to modules.

|
| It would need to be exported if is_global_init() was made static inline in a
| header (which seems like a sensible thing to do), but it wasn't.

It did not need to be exported in this patch.

I have a couple of follow-on patches that cleaned up some header-file
dependencies and made is_global_init() inline. Those patches are
changing a bit as I merge them with Pavel Emelianov's pid ns changes.

I will send a separate patch to inline is_global_init().

Suka

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