Subject: [PATCH 2.6.24-rc3-mm1] IPC: consolidate sem_exit_ns(), msg_exit_ns and shm_exit_ns()
Posted by Pierre Peiffer on Fri, 23 Nov 2007 16:52:50 GMT
View Forum Message <> Reply to Message

sem_exit_ns(), msg_exit_ns() and shm_exit_ns() are all called when an ipc_namespace is released to free all ipcs of each type.
But in fact, they do the same thing: they loop around all ipcs to free them individually by calling a specific routine.

This patch proposes to consolidate this by introducing a common function, free_ipcs(), that do the job. The specific routine to call on each individual ipcs is passed as parameter. For this, these ipc-specific 'free' routines are reworked to take a generic 'struct ipc_perm' as parameter.

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
---
 include/linux/ipc_namespace.h | 5 ++++-
 ipc/msg.c | 28 +++++-----------------
 ipc/namespace.c | 30 ++++++++++++++++++++++++++++++
 ipc/sem.c | 27 +++++----------------------
 ipc/shm.c | 27 +++++----------------------
 5 files changed, 50 insertions(+), 67 deletions(-)

Index: b/ipc/msg.c
===
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -72,7 +72,7 @@ struct msg_sender {
 #define msg_unlock(msq) ipc_unlock(&(msq)->q_perm)
 #define msg_buildid(id, seq) ipc_buildid(id, seq)

-static void freeque(struct ipc_namespace *, struct msg_queue *);
+static void freeque(struct ipc_namespace *, struct kern_ipc_perm *);
 static int newque(struct ipc_namespace *, struct ipc_params *);
 #ifdef CONFIG_PROC_FS
 static int sysvipc_msg_proc_show(struct seq_file *s, void *it);
@@ -91,26 +91,7 @@ void msg_init_ns(struct ipc_namespace *n
 #ifdef CONFIG_IPC_NS
 void msg_exit_ns(struct ipc_namespace *ns)
 {
- struct msg_queue *msq;
- struct kern_ipc_perm *perm;
- int next_id;
- int total, in_use;
-
- down_write(&msg_ids(ns).rw_mutex);
-

```
- in_use = msg_ids(ns).in_use;
-
- for (total = 0, next_id = 0; total < in_use; next_id++) {
-  perm = idr_find(&msg_ids(ns).ipcs_idr, next_id);
-  if (perm == NULL)
-   continue;
-  ipc_lock_by_ptr(perm);
-  msq = container_of(perm, struct msg_queue, q_perm);
-  freeque(ns, msq);
-  total++;
- }
-
- up_write(&msg_ids(ns).rw_mutex);
+ free_ipcs(ns, &msg_ids(ns), freeque);
 }
 #endif

@@ -274,9 +255,10 @@ static void expunge_all(struct msg_queue
  * msg_ids.rw_mutex (writer) and the spinlock for this message queue are held
  * before freeque() is called. msg_ids.rw_mutex remains locked on exit.
  */
-static void freeque(struct ipc_namespace *ns, struct msg_queue *msq)
+static void freeque(struct ipc_namespace *ns, struct kern_ipc_perm *ipcp)
 {
  struct list_head *tmp;
+ struct msg_queue *msq = container_of(ipcp, struct msg_queue, q_perm);

  expunge_all(msq, -EIDRM);
  ss_wakeup(&msq->q_senders, 1);
@@ -582,7 +564,7 @@ asmlinkage long sys_msgctl(int msqid, in
   break;
  }
  case IPC_RMID:
-  freeque(ns, msq);
+  freeque(ns, &msq->q_perm);
   break;
  }
  err = 0;
Index: b/ipc/namespace.c
===================================================================
--- a/ipc/namespace.c
+++ b/ipc/namespace.c
@@ -44,6 +44,36 @@ struct ipc_namespace *copy_ipcs(unsigned
 return new_ns;
}

+/*
+ * free_ipcs - free all ipcs of one type
```

```
+ * @ns:   the namespace to remove the ipcs from
+ * @ids:  the table of ipcs to free
+ * @free: the function called to free each individual ipc
+ *
+ * Called for each kind of ipc when an ipc_namespace exits.
+ */
+void free_ipcs(struct ipc_namespace *ns, struct ipc_ids *ids,
+        void (*free)(struct ipc_namespace *, struct kern_ipc_perm *))
+{
+ struct kern_ipc_perm *perm;
+ int next_id;
+ int total, in_use;
+
+ down_write(&ids->rw_mutex);
+
+ in_use = ids->in_use;
+
+ for (total = 0, next_id = 0; total < in_use; next_id++) {
+  perm = idr_find(&ids->ipcs_idr, next_id);
+  if (perm == NULL)
+   continue;
+  ipc_lock_by_ptr(perm);
+  free(ns, perm);
+  total++;
+ }
+ up_write(&ids->rw_mutex);
+}
+
 void free_ipc_ns(struct kref *kref)
 {
  struct ipc_namespace *ns;
Index: b/ipc/sem.c
===================================================================
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -94,7 +94,7 @@
 #define sem_buildid(id, seq) ipc_buildid(id, seq)

 static int newary(struct ipc_namespace *, struct ipc_params *);
-static void freeary(struct ipc_namespace *, struct sem_array *);
+static void freeary(struct ipc_namespace *, struct kern_ipc_perm *);
 #ifdef CONFIG_PROC_FS
 static int sysvipc_sem_proc_show(struct seq_file *s, void *it);
 #endif
@@ -129,25 +129,7 @@ void sem_init_ns(struct ipc_namespace *n
 #ifdef CONFIG_IPC_NS
 void sem_exit_ns(struct ipc_namespace *ns)
 {
```

```
- struct sem_array *sma;
- struct kern_ipc_perm *perm;
- int next_id;
- int total, in_use;
-
- down_write(&sem_ids(ns).rw_mutex);
-
- in_use = sem_ids(ns).in_use;
-
- for (total = 0, next_id = 0; total < in_use; next_id++) {
-  perm = idr_find(&sem_ids(ns).ipcs_idr, next_id);
-  if (perm == NULL)
-   continue;
-  ipc_lock_by_ptr(perm);
-  sma = container_of(perm, struct sem_array, sem_perm);
-  freeary(ns, sma);
-  total++;
- }
- up_write(&sem_ids(ns).rw_mutex);
+ free_ipcs(ns, &sem_ids(ns), freeary);
 }
 #endif

@@ -542,10 +524,11 @@ static int count_semzcnt (struct sem_arr
  * as a writer and the spinlock for this semaphore set hold. sem_ids.rw_mutex
  * remains locked on exit.
  */
-static void freeary(struct ipc_namespace *ns, struct sem_array *sma)
+static void freeary(struct ipc_namespace *ns, struct kern_ipc_perm *ipcp)
 {
  struct sem_undo *un;
  struct sem_queue *q;
+ struct sem_array *sma = container_of(ipcp, struct sem_array, sem_perm);

  /* Invalidate the existing undo structures for this semaphore set.
   * (They will be freed without any further action in exit_sem()
@@ -926,7 +909,7 @@ static int semctl_down(struct ipc_namesp

  switch(cmd){
  case IPC_RMID:
-  freeary(ns, sma);
+  freeary(ns, ipcp);
   err = 0;
   break;
  case IPC_SET:
Index: b/ipc/shm.c
===================================================================
--- a/ipc/shm.c
```

```
+++ b/ipc/shm.c
@@ -83,8 +83,11 @@ void shm_init_ns(struct ipc_namespace *n
  * Called with shm_ids.rw_mutex (writer) and the shp structure locked.
  * Only shm_ids.rw_mutex remains locked on exit.
  */
-static void do_shm_rmid(struct ipc_namespace *ns, struct shmid_kernel *shp)
+static void do_shm_rmid(struct ipc_namespace *ns, struct kern_ipc_perm *ipcp)
 {
+ struct shmid_kernel *shp;
+ shp = container_of(ipcp, struct shmid_kernel, shm_perm);
+
  if (shp->shm_nattch){
   shp->shm_perm.mode |= SHM_DEST;
   /* Do not find it any more */
@@ -97,25 +100,7 @@ static void do_shm_rmid(struct ipc_names
 #ifdef CONFIG_IPC_NS
 void shm_exit_ns(struct ipc_namespace *ns)
 {
- struct shmid_kernel *shp;
- struct kern_ipc_perm *perm;
- int next_id;
- int total, in_use;
-
- down_write(&shm_ids(ns).rw_mutex);
-
- in_use = shm_ids(ns).in_use;
-
- for (total = 0, next_id = 0; total < in_use; next_id++) {
-  perm = idr_find(&shm_ids(ns).ipcs_idr, next_id);
-  if (perm == NULL)
-   continue;
-  ipc_lock_by_ptr(perm);
-  shp = container_of(perm, struct shmid_kernel, shm_perm);
-  do_shm_rmid(ns, shp);
-  total++;
- }
- up_write(&shm_ids(ns).rw_mutex);
+ free_ipcs(ns, &shm_ids(ns), do_shm_rmid);
 }
 #endif

@@ -832,7 +817,7 @@ asmlinkage long sys_shmctl (int shmid, i
   if (err)
    goto out_unlock_up;

-  do_shm_rmid(ns, shp);
+  do_shm_rmid(ns, &shp->shm_perm);
   up_write(&shm_ids(ns).rw_mutex);
```

```
  goto out;
 }
Index: b/include/linux/ipc_namespace.h
===================================================================
--- a/include/linux/ipc_namespace.h
+++ b/include/linux/ipc_namespace.h
@@ -43,7 +43,10 @@ extern struct ipc_namespace init_ipc_ns;
 #if defined(CONFIG_SYSVIPC) && defined(CONFIG_IPC_NS)
 extern void free_ipc_ns(struct kref *kref);
 extern struct ipc_namespace *copy_ipcs(unsigned long flags,
-      struct ipc_namespace *ns);
+           struct ipc_namespace *ns);
+extern void free_ipcs(struct ipc_namespace *ns, struct ipc_ids *ids,
+        void (*free)(struct ipc_namespace *,
+       struct kern_ipc_perm *));

 static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
 {

--
Pierre Peiffer
```

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

---

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: consolidate sem_exit_ns(), msg_exit_ns and shm_exit_ns()
Posted by akpm on Tue, 27 Nov 2007 06:44:38 GMT
View Forum Message <> Reply to Message

On Fri, 23 Nov 2007 17:52:50 +0100 Pierre Peiffer <pierre.peiffer@bull.net> wrote:

> sem_exit_ns(), msg_exit_ns() and shm_exit_ns() are all called when an ipc_namespace is
> released to free all ipcs of each type.
> But in fact, they do the same thing: they loop around all ipcs to free them
> individually by calling a specific routine.
>
> This patch proposes to consolidate this by introducing a common function, free_ipcs(),
> that do the job. The specific routine to call on each individual ipcs is passed as
> parameter. For this, these ipc-specific 'free' routines are reworked to take a
> generic 'struct ipc_perm' as parameter.

This conflicts in more-than-trivial ways with Pavel's
move-the-ipc-namespace-under-ipc_ns-option.patch, which was in
2.6.24-rc3-mm1.

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: consolidate sem_exit_ns(), msg_exit_ns and shm_exit_ns()
Posted by akpm on Tue, 27 Nov 2007 06:46:36 GMT
View Forum Message <> Reply to Message

On Mon, 26 Nov 2007 22:44:38 -0800 Andrew Morton <akpm@linux-foundation.org> wrote:

> On Fri, 23 Nov 2007 17:52:50 +0100 Pierre Peiffer <pierre.peiffer@bull.net> wrote:
>
> > sem_exit_ns(), msg_exit_ns() and shm_exit_ns() are all called when an ipc_namespace is
> > released to free all ipcs of each type.
> > But in fact, they do the same thing: they loop around all ipcs to free them
> > individually by calling a specific routine.
> >
> > This patch proposes to consolidate this by introducing a common function, free_ipcs(),
> > that do the job. The specific routine to call on each individual ipcs is passed as
> > parameter. For this, these ipc-specific 'free' routines are reworked to take a
> > generic 'struct ipc_perm' as parameter.
>
> This conflicts in more-than-trivial ways with Pavel's
> move-the-ipc-namespace-under-ipc_ns-option.patch, which was in
> 2.6.24-rc3-mm1.
>

err, no, it wasn't that patch. For some reason your change assumes that
msg_exit_ns() (for example) doesn't have these lines:

```
	kfree(ns->ids[IPC_MSG_IDS]);
	ns->ids[IPC_MSG_IDS] = NULL;
```

in it.

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

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: consolidate sem_exit_ns(), msg_exit_ns and shm_exit_ns()
Posted by Pierre Peiffer on Tue, 27 Nov 2007 08:19:34 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Mon, 26 Nov 2007 22:44:38 -0800 Andrew Morton <akpm@linux-foundation.org> wrote:
>
>> On Fri, 23 Nov 2007 17:52:50 +0100 Pierre Peiffer <pierre.peiffer@bull.net> wrote:
>>
>>> sem_exit_ns(), msg_exit_ns() and shm_exit_ns() are all called when an ipc_namespace is
>>> released to free all ipcs of each type.
>>> But in fact, they do the same thing: they loop around all ipcs to free them
>>> individually by calling a specific routine.
>>>
>>> This patch proposes to consolidate this by introducing a common function, free_ipcs(),
>>> that do the job. The specific routine to call on each individual ipcs is passed as
>>> parameter. For this, these ipc-specific 'free' routines are reworked to take a
>>> generic 'struct ipc_perm' as parameter.
>> This conflicts in more-than-trivial ways with Pavel's
>> move-the-ipc-namespace-under-ipc_ns-option.patch, which was in
>> 2.6.24-rc3-mm1.
>>
>
> err, no, it wasn't that patch. For some reason your change assumes that
> msg_exit_ns() (for example) doesn't have these lines:
>
>   kfree(ns->ids[IPC_MSG_IDS]);
>   ns->ids[IPC_MSG_IDS] = NULL;
>
> in it.

Yes, in fact, I've made this patch on top of this one:
http://lkml.org/lkml/2007/11/22/49

As the patch mentioned by this previous thread was acked by Cedric and Pavel,
I've assumed that you will take both. But I've not made this clear, sorry.

--
Pierre

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

---

Subject: Re: [PATCH 2.6.24-rc3-mm1] IPC: consolidate sem_exit_ns(), msg_exit_ns and shm_exit_ns()
Posted by akpm on Tue, 27 Nov 2007 08:29:22 GMT
View Forum Message <> Reply to Message

On Tue, 27 Nov 2007 09:19:34 +0100 Pierre Peiffer <pierre.peiffer@bull.net> wrote:

>
>
> Andrew Morton wrote:
> > On Mon, 26 Nov 2007 22:44:38 -0800 Andrew Morton <akpm@linux-foundation.org> wrote:
> >
> >> On Fri, 23 Nov 2007 17:52:50 +0100 Pierre Peiffer <pierre.peiffer@bull.net> wrote:
> >>
> >>> sem_exit_ns(), msg_exit_ns() and shm_exit_ns() are all called when an ipc_namespace is
> >>> released to free all ipcs of each type.
> >>> But in fact, they do the same thing: they loop around all ipcs to free them
> >>> individually by calling a specific routine.
> >>>
> >>> This patch proposes to consolidate this by introducing a common function, free_ipcs(),
> >>> that do the job. The specific routine to call on each individual ipcs is passed as
> >>> parameter. For this, these ipc-specific 'free' routines are reworked to take a
> >>> generic 'struct ipc_perm' as parameter.
> >> This conflicts in more-than-trivial ways with Pavel's
> >> move-the-ipc-namespace-under-ipc_ns-option.patch, which was in
> >> 2.6.24-rc3-mm1.
> >>
> >
> > err, no, it wasn't that patch. For some reason your change assumes that
> > msg_exit_ns() (for example) doesn't have these lines:
> >
> >   kfree(ns->ids[IPC_MSG_IDS]);
> >   ns->ids[IPC_MSG_IDS] = NULL;
> >
> > in it.
>
> Yes, in fact, I've made this patch on top of this one:
> http://lkml.org/lkml/2007/11/22/49
>
> As the patch mentioned by this previous thread was acked by Cedric and Pavel,
> I've assumed that you will take both.

doh, I misread the discussion and assumed that a new version was due, sorry.

> But I've not made this clear, sorry.

Well, sequence-numbering the patches as

[patch 2/5] ipc: <stuff>
[patch 5/5] ipc: <more stuff>

always helps. Emails get reordered in flight, but more importantly this
numbering helps ensure that none of the patches get lost.
