Subject: Re: [PATCH] Consolidate IPC namespace cleanup functions
Posted by Cedric Le Goater on Tue, 02 Oct 2007 15:01:41 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov wrote:
> When the IPC namespace is terminated all the IPC objects (i.e. ids)
> living in it are freed. This is done in a similar way in X_exit_ns()
> functions. All the code can be consolidated, saving 122 bytes when
> the NAMESPACES are on.
>
> This patch must be applied after the ones with the NAMESPACES config
> option introduced.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Acked-by: Cedric Le Goater <clg@fr.ibm.com>

Thanks Pavel !

C.

>
> ---
>
> ipc/msg.c | 23 ++++----------------
> ipc/namespace.c | 39 ++++++++++++++++++++++++++++++++++++
> ipc/sem.c | 23 ++++----------------
> ipc/shm.c | 23 ++++----------------
> ipc/util.h | 6 +++---
> 5 files changed, 54 insertions(+), 60 deletions(-)
>
> diff --git a/ipc/namespace.c b/ipc/namespace.c
> index cef1139..98de4e5 100644
> --- a/ipc/namespace.c
> +++ b/ipc/namespace.c
> @@ -12,6 +12,45 @@
>
> #include "util.h"
>
> +static void ipc_exit_ns(struct ipc_namespace *ns, struct ipc_ids *ids,
> + void (*free_fn)(struct ipc_namespace *ns, void *id))
> +{
> + void *id;
> + int next_id;
> + int total, in_use;
> +

```
> + mutex_lock(&ids->mutex);
> +
> + in_use = ids->in_use;
> +
> + for (total = 0, next_id = 0; total < in_use; next_id++) {
> +  id = idr_find(&ids->ipcs_idr, next_id);
> +  if (id == NULL)
> +   continue;
> +
> +  free_fn(ns, id);
> +  total++;
> + }
> + mutex_unlock(&ids->mutex);
> +
> + kfree(ids);
> +}
> +
> +static inline void sem_exit_ns(struct ipc_namespace *ns)
> +{
> + ipc_exit_ns(ns, ns->ids[IPC_SEM_IDS], sem_free);
> +}
> +
> +static inline void msg_exit_ns(struct ipc_namespace *ns)
> +{
> + ipc_exit_ns(ns, ns->ids[IPC_MSG_IDS], msg_free);
> +}
> +
> +static inline void shm_exit_ns(struct ipc_namespace *ns)
> +{
> + ipc_exit_ns(ns, ns->ids[IPC_SHM_IDS], shm_free);
> +}
> +
>  static struct ipc_namespace *clone_ipc_ns(struct ipc_namespace *old_ns)
>  {
>   int err;
> diff --git a/ipc/sem.c b/ipc/sem.c
> index 2e9f449..8027a30 100644
> --- a/ipc/sem.c
> +++ b/ipc/sem.c
> @@ -144,28 +144,13 @@ int sem_init_ns(struct ipc_namespace *ns
>   return 0;
>  }
>
> -void sem_exit_ns(struct ipc_namespace *ns)
> +void sem_free(struct ipc_namespace *ns, void *id)
>  {
>   struct sem_array *sma;
> - int next_id;
```

> - int total, in_use;
>
> - mutex_lock(&sem_ids(ns).mutex);
> -
> - in_use = sem_ids(ns).in_use;
> -
> - for (total = 0, next_id = 0; total < in_use; next_id++) {
> - sma = idr_find(&sem_ids(ns).ipcs_idr, next_id);
> - if (sma == NULL)
> -   continue;
> - ipc_lock_by_ptr(&sma->sem_perm);
> - freeary(ns, sma);
> - total++;
> - }
> - mutex_unlock(&sem_ids(ns).mutex);
> -
> - kfree(ns->ids[IPC_SEM_IDS]);
> - ns->ids[IPC_SEM_IDS] = NULL;
> + sma = (struct sem_array *)id;
> + ipc_lock_by_ptr(&sma->sem_perm);
> + freeary(ns, sma);
> }
> #endif
>
> diff --git a/ipc/msg.c b/ipc/msg.c
> index eb74965..9b8a155 100644
> --- a/ipc/msg.c
> +++ b/ipc/msg.c
> @@ -106,28 +106,13 @@ int msg_init_ns(struct ipc_namespace *ns
>   return 0;
> }
>
> -void msg_exit_ns(struct ipc_namespace *ns)
> +void msg_free(struct ipc_namespace *ns, void *id)
> {
>   struct msg_queue *msq;
> - int next_id;
> - int total, in_use;
>
> - mutex_lock(&msg_ids(ns).mutex);
> -
> - in_use = msg_ids(ns).in_use;
> -
> - for (total = 0, next_id = 0; total < in_use; next_id++) {
> - msq = idr_find(&msg_ids(ns).ipcs_idr, next_id);
> - if (msq == NULL)
> -   continue;
> - ipc_lock_by_ptr(&msq->q_perm);

```
> -  freeque(ns, msq);
> -  total++;
> - }
> - mutex_unlock(&msg_ids(ns).mutex);
> -
> - kfree(ns->ids[IPC_MSG_IDS]);
> - ns->ids[IPC_MSG_IDS] = NULL;
> + msq = (struct msg_queue *)id;
> + ipc_lock_by_ptr(&msq->q_perm);
> + freeque(ns, msq);
>  }
>  #endif
>
> diff --git a/ipc/shm.c b/ipc/shm.c
> index 2717cbc..8f50166 100644
> --- a/ipc/shm.c
> +++ b/ipc/shm.c
> @@ -111,28 +111,13 @@ int shm_init_ns(struct ipc_namespace *ns
>   return 0;
>  }
>
> -void shm_exit_ns(struct ipc_namespace *ns)
> +void shm_free(struct ipc_namespace *ns, void *id)
>  {
>   struct shmid_kernel *shp;
> - int next_id;
> - int total, in_use;
> -
> - mutex_lock(&shm_ids(ns).mutex);
> -
> - in_use = shm_ids(ns).in_use;
> -
> - for (total = 0, next_id = 0; total < in_use; next_id++) {
> -  shp = idr_find(&shm_ids(ns).ipcs_idr, next_id);
> -  if (shp == NULL)
> -   continue;
> -  ipc_lock_by_ptr(&shp->shm_perm);
> -  do_shm_rmid(ns, shp);
> -  total++;
> - }
> - mutex_unlock(&shm_ids(ns).mutex);
>
> - kfree(ns->ids[IPC_SHM_IDS]);
> - ns->ids[IPC_SHM_IDS] = NULL;
> + shp = (struct shmid_kernel *)id;
> + ipc_lock_by_ptr(&shp->shm_perm);
> + do_shm_rmid(ns, shp);
>  }
```

> #endif
>
> diff --git a/ipc/util.h b/ipc/util.h
> index 8972402..f1f0a31 100644
> --- a/ipc/util.h
> +++ b/ipc/util.h
> @@ -26,9 +26,9 @@ int sem_init_ns(struct ipc_namespace *ns
> int msg_init_ns(struct ipc_namespace *ns);
> int shm_init_ns(struct ipc_namespace *ns);
>
> -void sem_exit_ns(struct ipc_namespace *ns);
> -void msg_exit_ns(struct ipc_namespace *ns);
> -void shm_exit_ns(struct ipc_namespace *ns);
> +void sem_free(struct ipc_namespace *ns, void *id);
> +void msg_free(struct ipc_namespace *ns, void *id);
> +void shm_free(struct ipc_namespace *ns, void *id);
>
> struct ipc_ids {
>   int in_use;