
Subject: [PATCH 3/6] Move the IPC namespace under IPC_NS option
Posted by [Pavel Emelianov](#) on Wed, 14 Nov 2007 11:32:43 GMT
[View Forum Message](#) <> [Reply to Message](#)

Currently the IPC namespace management code is spread over the ipc/*.c files. I moved this code into ipc/namespace.c file which is compiled out when needed.

The linux/ipc_namespace.h file is used to store the prototypes of the functions in namespace.c and the stubs for NAMESPACES=n case. This is done so, because the stub for copy_ipc_namespace requires the knowledge of the CLONE_NEWIPC flag, which is in sched.h. But the linux/ipc.h file itself is included into many many .c files via the sys.h->sem.h sequence so adding the sched.h into it will make all these .c depend on sched.h which is not that good. On the other hand the knowledge about the namespaces stuff is required in 4 .c files only.

Besides, this patch compiles out some auxiliary functions from ipc/sem.c, msg.c and shm.c files. It turned out that moving these functions into namespaces.c is not that easy because they use many other calls and macros from the original file. Moving them would make this patch complicated. On the other hand all these functions can be consolidated, so I will send a separate patch doing this a bit later.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
diff --git a/include/linux/ipc.h b/include/linux/ipc.h
index 408696e..b882610 100644
```

```
--- a/include/linux/ipc.h
```

```
+++ b/include/linux/ipc.h
```

```
@@ -100,58 +100,6 @@ struct kern_ipc_perm
    void *security;
};
```

```
-struct ipc_ids;
```

```
-struct ipc_namespace {
```

```
- struct kref kref;
```

```
- struct ipc_ids *ids[3];
```

```
-
```

```
- int sem_ctls[4];
```

```
- int used_sems;
```

```
-
```

```
- int msg_ctlmax;
```

```

- int msg_ctlmnb;
- int msg_ctlmni;
- atomic_t msg_bytes;
- atomic_t msg_hdrs;
-
- size_t shm_ctlmax;
- size_t shm_ctlall;
- int shm_ctlmni;
- int shm_tot;
-};
-
-extern struct ipc_namespace init_ipc_ns;
-
-#ifdef CONFIG_SYSVIPC
-#define INIT_IPC_NS(ns) .ns = &init_ipc_ns,
-extern void free_ipc_ns(struct kref *kref);
-extern struct ipc_namespace *copy_ipcs(unsigned long flags,
-    struct ipc_namespace *ns);
-#else
-#define INIT_IPC_NS(ns)
-static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
-    struct ipc_namespace *ns)
-{
- return ns;
-}
-#endif
-
-static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
-{
-#ifdef CONFIG_SYSVIPC
- if (ns)
- kref_get(&ns->kref);
-#endif
- return ns;
-}
-
-static inline void put_ipc_ns(struct ipc_namespace *ns)
-{
-#ifdef CONFIG_SYSVIPC
- kref_put(&ns->kref, free_ipc_ns);
-#endif
-}
-
-#endif /* __KERNEL__ */

#endif /* _LINUX_IPC_H */
diff --git a/include/linux/ipc_namespace.h b/include/linux/ipc_namespace.h
new file mode 100644

```

```

index 0000000..a491fc9
--- /dev/null
+++ b/include/linux/ipc_namespace.h
@@ -0,0 +1,69 @@
+#ifndef __IPC_NAMESPACE_H__
+#define __IPC_NAMESPACE_H__
+
+#include <linux/err.h>
+
+struct ipc_ids;
+struct ipc_namespace {
+ struct kref kref;
+ struct ipc_ids *ids[3];
+
+ int sem_ctls[4];
+ int used_sems;
+
+ int msg_ctlmax;
+ int msg_ctlmnb;
+ int msg_ctlmni;
+ atomic_t msg_bytes;
+ atomic_t msg_hdrs;
+
+ size_t shm_ctlmax;
+ size_t shm_ctlall;
+ int shm_ctlmni;
+ int shm_tot;
+};
+
+extern struct ipc_namespace init_ipc_ns;
+
+#ifdef CONFIG_SYSVIPC
+#define INIT_IPC_NS(ns) .ns = &init_ipc_ns,
+#else
+#define INIT_IPC_NS(ns)
+#endif
+
+#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);
+
+static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
+{
+ if (ns)
+ kref_get(&ns->kref);
+ return ns;
+}

```



```

obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
obj_mq-$(CONFIG_COMPAT) += compat_mq.o
obj-$(CONFIG_POSIX_QUEUE) += mqueue.o msgutil.o $(obj_mq-y)
+obj-$(CONFIG_IPC_NS) += namespace.o

diff --git a/ipc/ipc_sysctl.c b/ipc/ipc_sysctl.c
index 79e24e8..7f4235b 100644
--- a/ipc/ipc_sysctl.c
+++ b/ipc/ipc_sysctl.c
@@ -14,6 +14,7 @@
#include <linux/nsproxy.h>
#include <linux/sysctl.h>
#include <linux/uaccess.h>
+#include <linux/ipc_namespace.h>

static void *get_ipc(ctl_table *table)
{
diff --git a/ipc/msg.c b/ipc/msg.c
index ec0c724..5879bfe 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -36,6 +36,7 @@
#include <linux/seq_file.h>
#include <linux/rwsem.h>
#include <linux/nsproxy.h>
+#include <linux/ipc_namespace.h>

#include <asm/current.h>
#include <asm/uaccess.h>
@@ -90,6 +91,7 @@ static void __msg_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
ipc_init_ids(ids);
}

+#ifdef CONFIG_IPC_NS
int msg_init_ns(struct ipc_namespace *ns)
{
struct ipc_ids *ids;
@@ -128,6 +130,7 @@ void msg_exit_ns(struct ipc_namespace *ns)
kfree(ns->ids[IPC_MSG_IDS]);
ns->ids[IPC_MSG_IDS] = NULL;
}
+#endif

void __init msg_init(void)
{
diff --git a/ipc/namespace.c b/ipc/namespace.c
new file mode 100644
index 0000000..cef1139

```

```

--- /dev/null
+++ b/ipc/namespace.c
@@ -0,0 +1,73 @@
+/*
+ * linux/ipc/namespace.c
+ * Copyright (C) 2006 Pavel Emelyanov <xemul@openvz.org> OpenVZ, SWsoft Inc.
+ */
+
+#include <linux/ipc.h>
+#include <linux/msg.h>
+#include <linux/ipc_namespace.h>
+#include <linux/rcupdate.h>
+#include <linux/nsproxy.h>
+#include <linux/slab.h>
+
+#include "util.h"
+
+static struct ipc_namespace *clone_ipc_ns(struct ipc_namespace *old_ns)
+{
+ int err;
+ struct ipc_namespace *ns;
+
+ err = -ENOMEM;
+ ns = kmalloc(sizeof(struct ipc_namespace), GFP_KERNEL);
+ if (ns == NULL)
+ goto err_mem;
+
+ err = sem_init_ns(ns);
+ if (err)
+ goto err_sem;
+ err = msg_init_ns(ns);
+ if (err)
+ goto err_msg;
+ err = shm_init_ns(ns);
+ if (err)
+ goto err_shm;
+
+ kref_init(&ns->kref);
+ return ns;
+
+err_shm:
+ msg_exit_ns(ns);
+err_msg:
+ sem_exit_ns(ns);
+err_sem:
+ kfree(ns);
+err_mem:
+ return ERR_PTR(err);

```

```

+}
+
+struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
+{
+ struct ipc_namespace *new_ns;
+
+ BUG_ON(!ns);
+ get_ipc_ns(ns);
+
+ if (!(flags & CLONE_NEWIPC))
+ return ns;
+
+ new_ns = clone_ipc_ns(ns);
+
+ put_ipc_ns(ns);
+ return new_ns;
+}
+
+void free_ipc_ns(struct kref *kref)
+{
+ struct ipc_namespace *ns;
+
+ ns = container_of(kref, struct ipc_namespace, kref);
+ sem_exit_ns(ns);
+ msg_exit_ns(ns);
+ shm_exit_ns(ns);
+ kfree(ns);
+}
diff --git a/ipc/sem.c b/ipc/sem.c
index d65e285..84c701f 100644
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -82,6 +82,7 @@
#include <linux/seq_file.h>
#include <linux/rwsem.h>
#include <linux/nsproxy.h>
+#include <linux/ipc_namespace.h>

#include <asm/uaccess.h>
#include "util.h"
@@ -128,6 +129,7 @@ static void __sem_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
ipc_init_ids(ids);
}

#ifdef CONFIG_IPC_NS
int sem_init_ns(struct ipc_namespace *ns)
{
struct ipc_ids *ids;

```

```

@@ -165,6 +167,7 @@ void sem_exit_ns(struct ipc_namespace *ns)
    kfree(ns->ids[IPC_SEM_IDS]);
    ns->ids[IPC_SEM_IDS] = NULL;
}
+#endif

```

```

void __init sem_init (void)
{
diff --git a/ipc/shm.c b/ipc/shm.c
index 65c3a29..07f4b7a 100644
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -38,6 +38,7 @@
#include <linux/rwsem.h>
#include <linux/nsproxy.h>
#include <linux/mount.h>
+#include <linux/ipc_namespace.h>

```

```

#include <asm/uaccess.h>

```

```

@@ -96,6 +97,7 @@ static void do_shm_rmid(struct ipc_namespace *ns, struct shmid_kernel
*shp)
    shm_destroy(ns, shp);
}

```

```

+#ifdef CONFIG_IPC_NS
int shm_init_ns(struct ipc_namespace *ns)
{
    struct ipc_ids *ids;
@@ -133,6 +135,7 @@ void shm_exit_ns(struct ipc_namespace *ns)
    kfree(ns->ids[IPC_SHM_IDS]);
    ns->ids[IPC_SHM_IDS] = NULL;
}
+#endif

```

```

void __init shm_init (void)
{
diff --git a/ipc/util.c b/ipc/util.c
index 76c1f34..5432b8e 100644
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -33,6 +33,7 @@
#include <linux/audit.h>
#include <linux/nsproxy.h>
#include <linux/rwsem.h>
+#include <linux/ipc_namespace.h>

#include <asm/unistd.h>

```



```

@@ -51,66 +52,6 @@ struct ipc_namespace init_ipc_ns = {
    },
};

-static struct ipc_namespace *clone_ipc_ns(struct ipc_namespace *old_ns)
-{
- int err;
- struct ipc_namespace *ns;
-
- err = -ENOMEM;
- ns = kmalloc(sizeof(struct ipc_namespace), GFP_KERNEL);
- if (ns == NULL)
- goto err_mem;
-
- err = sem_init_ns(ns);
- if (err)
- goto err_sem;
- err = msg_init_ns(ns);
- if (err)
- goto err_msg;
- err = shm_init_ns(ns);
- if (err)
- goto err_shm;
-
- kref_init(&ns->kref);
- return ns;
-
-err_shm:
- msg_exit_ns(ns);
-err_msg:
- sem_exit_ns(ns);
-err_sem:
- kfree(ns);
-err_mem:
- return ERR_PTR(err);
-}
-
-struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
-{
- struct ipc_namespace *new_ns;
-
- BUG_ON(!ns);
- get_ipc_ns(ns);
-
- if (!(flags & CLONE_NEWIPC))
- return ns;
-
-

```

```

- new_ns = clone_ipc_ns(ns);
-
- put_ipc_ns(ns);
- return new_ns;
-}
-
-void free_ipc_ns(struct kref *kref)
-{
- struct ipc_namespace *ns;
-
- ns = container_of(kref, struct ipc_namespace, kref);
- sem_exit_ns(ns);
- msg_exit_ns(ns);
- shm_exit_ns(ns);
- kfree(ns);
-}

```

```

/**

```

```

 * ipc_init - initialise IPC subsystem

```

```

 *

```

```

diff --git a/ipc/util.h b/ipc/util.h
index 9ffea40..fc6b729 100644
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -20,6 +20,8 @@ void sem_init (void);
void msg_init (void);
void shm_init (void);

```

```

+struct ipc_namespace;
+
+int sem_init_ns(struct ipc_namespace *ns);
+int msg_init_ns(struct ipc_namespace *ns);
+int shm_init_ns(struct ipc_namespace *ns);
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index 79f871b..f5d332c 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -21,6 +21,7 @@
#include <linux/utsname.h>
#include <linux/pid_namespace.h>
#include <net/net_namespace.h>
+#include <linux/ipc_namespace.h>

```

```

static struct kmem_cache *nsproxy_cache;

```

```

--

```

1.5.3.4

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