Subject: [RFC][PATCH 2/2] Virtualization of IPC
Posted by dev on Fri, 24 Mar 2006 17:35:58 GMT
View Forum Message <> Reply to Message

This patch introduces IPC namespaces, which allow to create isolated IPC
users or containers.
Introduces CONFIG_IPC_NS and ipc_namespace structure.
It also uses current->ipc_ns as a pointer to current namespace, which
reduces places where additional argument to functions should be added.

Signed-Off-By: Pavel Emelianov <xemul@openvz.org>
Signed-Off-By: Kirill Korotaev <dev@openvz.org>

From: Pavel Emelianov <xemul@openvz.org>
Date: Wed, 22 Mar 2006 13:15:23 +0000 (-0500)
Subject: Move ipc objects into separate namespace.
X-Git-Url:
http://git.openvz.org/?p=linux-2.6-openvz-ms;a=commitdiff;h=84a441429e7af425d27a8cda5beb7
0de5521a0be

Move ipc objects into separate namespace.

```
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -4,6 +4,7 @@
 #include <linux/file.h>
 #include <linux/rcupdate.h>
 #include <linux/utsname.h>
+#include <linux/ipc.h>

 #define INIT_FDTABLE \
 {      \
@@ -79,6 +80,12 @@ extern struct group_info init_groups;
 #define INIT_UTS_NS
 #endif

+#ifdef CONFIG_IPC_NS
+#define INIT_IPC_NS .ipc_ns = &init_ipc_ns,
+#else
+#define INIT_IPC_NS
+#endif
+
 /*
  *  INIT_TASK is used to set up the first task table, touch at
  * your own risk!. Base=0, limit=0x1fffff (=2MB)
@@ -129,6 +136,7 @@ extern struct group_info init_groups;
  .cpu_timers = INIT_CPU_TIMERS(tsk.cpu_timers),  \
```

```
  .fs_excl = ATOMIC_INIT(0),    \
  INIT_UTS_NS       \
+ INIT_IPC_NS       \
 }


--- a/include/linux/ipc.h
+++ b/include/linux/ipc.h
@@ -70,6 +70,50 @@ struct kern_ipc_perm

 #endif /* __KERNEL__ */

+#include <linux/config.h>
+
+#ifdef CONFIG_IPC_NS
+#include <asm/atomic.h>
+
+struct ipc_ids;
+struct ipc_namespace {
+ atomic_t cnt;
+
+ struct ipc_ids *sem_ids;
+ int sem_ctls[4];
+ int used_sems;
+
+ struct ipc_ids *msg_ids;
+ int msg_ctlmax;
+ int msg_ctlmnb;
+ int msg_ctlmni;
+
+ struct ipc_ids *shm_ids;
+ size_t shm_ctlmax;
+ size_t  shm_ctlall;
+ int  shm_ctlmni;
+ int shm_total;
+};
+
+extern struct ipc_namespace init_ipc_ns;
+struct ipc_namespace *create_ipc_ns(void);
+void free_ipc_ns(struct ipc_namespace *ns);
+
+static inline void get_ipc_ns(struct ipc_namespace *ns)
+{
+ atomic_inc(&ns->cnt);
+}
+
+static inline void put_ipc_ns(struct ipc_namespace *ns)
+{
```

```
+ if (atomic_dec_and_test(&ns->cnt))
+  free_ipc_ns(ns);
+}
+#else
+#define get_ipc_ns(ns) do { } while (0)
+#define put_ipc_ns(ns) do { } while (0)
+#endif
+
 #endif /* _LINUX_IPC_H */


--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -689,6 +689,7 @@ static inline void prefetch_stack(struct
 struct audit_context;  /* See audit.c */
 struct mempolicy;
 struct uts_namespace;
+struct ipc_namespace;

 struct task_struct {
  volatile long state; /* -1 unrunnable, 0 runnable, >0 stopped */
@@ -806,6 +807,9 @@ struct task_struct {
 #ifdef CONFIG_UTS_NS
  struct uts_namespace *uts_ns;
 #endif
+#ifdef CONFIG_IPC_NS
+ struct ipc_namespace *ipc_ns;
+#endif
 /* signal handlers */
  struct signal_struct *signal;
  struct sighand_struct *sighand;
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -32,11 +32,6 @@
 #include <asm/uaccess.h>
 #include "util.h"

-/* sysctl: */
-int msg_ctlmax = MSGMAX;
-int msg_ctlmnb = MSGMNB;
-int msg_ctlmni = MSGMNI;
-
 /* one msg_receiver structure for each sleeping receiver */
 struct msg_receiver {
  struct list_head r_list;
@@ -63,8 +58,6 @@ struct msg_sender {
 static atomic_t msg_bytes = ATOMIC_INIT(0);
 static atomic_t msg_hdrs = ATOMIC_INIT(0);
```

```
-static struct ipc_ids msg_ids;
-
 #define msg_lock(id) ((struct msg_queue*)ipc_lock(&msg_ids,id))
 #define msg_unlock(msq) ipc_unlock(&(msq)->q_perm)
 #define msg_rmid(id) ((struct msg_queue*)ipc_rmid(&msg_ids,id))
@@ -79,8 +72,46 @@ static int newque (key_t key, int msgflg
 static int sysvipc_msg_proc_show(struct seq_file *s, void *it);
 #endif

+static struct ipc_ids msg_ids;
+
+#ifndef CONFIG_IPC_NS
+int msg_ctlmax = MSGMAX;
+int msg_ctlmnb = MSGMNB;
+int msg_ctlmni = MSGMNI;
+#else
+int init_msg_ns(struct ipc_namespace *ns)
+{
+ struct ipc_ids *ids;
+
+ ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
+ if (ids == NULL)
+  return -ENOMEM;
+
+ ns->msg_ctlmax = MSGMAX;
+ ns->msg_ctlmnb = MSGMNB;
+ ns->msg_ctlmni = MSGMNI;
+ ns->msg_ids = ids;
+ ids->ns = ns;
+ ipc_init_ids(ids, ns->msg_ctlmni);
+ return 0;
+}
+
+void fini_msg_ns(struct ipc_namespace *ns)
+{
+ kfree(ns->msg_ids);
+}
+
+#define msg_ctlmax (current->ipc_ns->msg_ctlmax)
+#define msg_ctlmnb (current->ipc_ns->msg_ctlmnb)
+#define msg_ctlmni (current->ipc_ns->msg_ctlmni)
+#endif
+
 void __init msg_init (void)
 {
+#ifdef CONFIG_IPC_NS
+ init_ipc_ns.msg_ids = &msg_ids;
```

```
+ msg_ids.ns = &init_ipc_ns;
+#endif
  ipc_init_ids(&msg_ids,msg_ctlmni);
  ipc_init_proc_interface("sysvipc/msg",
     "       key      msqid perms      cbytes       qnum lspid lrpid   uid   gid  cuid  cgid      stime      rtime
     ctime\n",
@@ -88,6 +119,10 @@ void __init msg_init (void)
     sysvipc_msg_proc_show);
 }

+#ifdef CONFIG_IPC_NS
+#define msg_ids  (*(current->ipc_ns->msg_ids))
+#endif
+
 static int newque (key_t key, int msgflg)
 {
  int id;
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -86,7 +86,6 @@
  ipc_checkid(&sem_ids,&sma->sem_perm,semid)
 #define sem_buildid(id, seq) \
  ipc_buildid(&sem_ids, id, seq)
-static struct ipc_ids sem_ids;

 static int newary (key_t, int, int);
 static void freeary (struct sem_array *sma, int id);
@@ -106,17 +105,52 @@ static int sysvipc_sem_proc_show(struct
  *
  */

+static struct ipc_ids sem_ids;
+#ifndef CONFIG_IPC_NS
 int sem_ctls[4] = {SEMMSL, SEMMNS, SEMOPM, SEMMNI};
 #define sc_semmsl (sem_ctls[0])
 #define sc_semmns (sem_ctls[1])
 #define sc_semopm (sem_ctls[2])
 #define sc_semmni (sem_ctls[3])
-
 static int used_sems;
+#else
+int init_sem_ns(struct ipc_namespace *ns)
+{
+ struct ipc_ids *ids;
+
+ ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
+ if (ids == NULL)
+  return -ENOMEM;
```

```
+
+ ns->sem_ctls[0] = SEMMSL;
+ ns->sem_ctls[1] = SEMMNS;
+ ns->sem_ctls[2] = SEMOPM;
+ ns->sem_ctls[3] = SEMMNI;
+ ns->used_sems = 0;
+ ns->sem_ids = ids;
+ ids->ns = ns;
+ ipc_init_ids(ids, ns->sem_ctls[3]);
+ return 0;
+}
+
+void fini_sem_ns(struct ipc_namespace *ns)
+{
+ kfree(ns->sem_ids);
+}
+
+#define sc_semmsl (current->ipc_ns->sem_ctls[0])
+#define sc_semmns (current->ipc_ns->sem_ctls[1])
+#define sc_semopm (current->ipc_ns->sem_ctls[2])
+#define sc_semmni (current->ipc_ns->sem_ctls[3])
+#define used_sems (current->ipc_ns->used_sems)
+#endif

 void __init sem_init (void)
 {
- used_sems = 0;
+#ifdef CONFIG_IPC_NS
+ init_ipc_ns.sem_ids = &sem_ids;
+ sem_ids.ns = &init_ipc_ns;
+#endif
  ipc_init_ids(&sem_ids,sc_semmni);
  ipc_init_proc_interface("sysvipc/sem",
     "       key      semid perms      nsems   uid   gid  cuid  cgid      otime      ctime\n",
@@ -124,6 +158,10 @@ void __init sem_init (void)
     sysvipc_sem_proc_show);
 }

+#ifdef CONFIG_IPC_NS
+#define sem_ids  (*(current->ipc_ns->sem_ids))
+#endif
+
 /*
  * Lockless wakeup algorithm:
  * Without the check/retry algorithm a lockless wakeup is possible:
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -38,8 +38,6 @@
```

```
 static struct file_operations shm_file_operations;
 static struct vm_operations_struct shm_vm_ops;

-static struct ipc_ids shm_ids;
-
 #define shm_lock(id) ((struct shmid_kernel*)ipc_lock(&shm_ids,id))
 #define shm_unlock(shp) ipc_unlock(&(shp)->shm_perm)
 #define shm_get(id) ((struct shmid_kernel*)ipc_get(&shm_ids,id))
@@ -53,14 +51,48 @@ static void shm_close (struct vm_area_st
 static int sysvipc_shm_proc_show(struct seq_file *s, void *it);
 #endif

+static struct ipc_ids shm_ids;
+#ifndef CONFIG_IPC_NS
 size_t shm_ctlmax = SHMMAX;
 size_t  shm_ctlall = SHMALL;
 int  shm_ctlmni = SHMMNI;
+static int shm_total; /* total number of shared memory pages */
+#else
+int init_shm_ns(struct ipc_namespace *ns)
+{
+ struct ipc_ids *ids;

-static int shm_tot; /* total number of shared memory pages */
+ ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
+ if (ids == NULL)
+  return -ENOMEM;
+
+ ns->shm_ctlmax = SHMMAX;
+ ns->shm_ctlall = SHMALL;
+ ns->shm_ctlmni = SHMMNI;
+ ns->shm_total = 0;
+ ns->shm_ids = ids;
+ ids->ns = ns;
+ ipc_init_ids(ids, 1);
+ return 0;
+}
+
+void fini_shm_ns(struct ipc_namespace *ns)
+{
+ kfree(ns->shm_ids);
+}
+
+#define shm_ctlmax (current->ipc_ns->shm_ctlmax)
+#define shm_ctlall (current->ipc_ns->shm_ctlall)
+#define shm_ctlmni (current->ipc_ns->shm_ctlmni)
+#define shm_total (current->ipc_ns->shm_total)
+#endif
```

```
 void __init shm_init (void)
 {
+#ifdef CONFIG_IPC_NS
+ init_ipc_ns.shm_ids = &shm_ids;
+ shm_ids.ns = &init_ipc_ns;
+#endif
  ipc_init_ids(&shm_ids, 1);
  ipc_init_proc_interface("sysvipc/shm",
     "       key      shmid perms       size  cpid  lpid nattch   uid   gid  cuid  cgid     atime      dtime
ctime\n",
@@ -68,6 +100,10 @@ void __init shm_init (void)
     sysvipc_shm_proc_show);
 }

+#ifdef CONFIG_IPC_NS
+#define shm_ids  (*(current->ipc_ns->shm_ids))
+#endif
+
 static inline int shm_checkid(struct shmid_kernel *s, int id)
 {
  if (ipc_checkid(&shm_ids,&s->shm_perm,id))
@@ -114,7 +150,15 @@ static void shm_open (struct vm_area_str
  */
 static void shm_destroy (struct shmid_kernel *shp)
 {
- shm_tot -= (shp->shm_segsz + PAGE_SIZE - 1) >> PAGE_SHIFT;
+#ifdef CONFIG_IPC_NS
+ struct ipc_namespace *ns, *cur;
+
+ ns = (struct ipc_namespace *)shp->shm_file->private_data;
+ cur = current->ipc_ns;
+ current->ipc_ns = ns;
+#endif
+
+ shm_total -= (shp->shm_segsz + PAGE_SIZE - 1) >> PAGE_SHIFT;
  shm_rmid (shp->id);
  shm_unlock(shp);
  if (!is_file_hugepages(shp->shm_file))
@@ -125,6 +169,10 @@ static void shm_destroy (struct shmid_ke
  fput (shp->shm_file);
  security_shm_free(shp);
  ipc_rcu_putref(shp);
+
+#ifdef CONFIG_IPC_NS
+ current->ipc_ns = cur;
+#endif
 }
```

```
 /*
@@ -167,8 +215,23 @@ static int shm_mmap(struct file * file,
  return ret;
 }

+#ifdef CONFIG_IPC_NS
+static int shm_release(struct inode *ino, struct file *file)
+{
+ struct ipc_namespace *ns;
+
+ ns = (struct ipc_namespace *)file->private_data;
+ file->private_data = NULL;
+ put_ipc_ns(ns);
+ return 0;
+}
+#endif
+
 static struct file_operations shm_file_operations = {
- .mmap = shm_mmap,
+ .mmap  = shm_mmap,
+#ifdef CONFIG_IPC_NS
+ .release = shm_release,
+#endif
 #ifndef CONFIG_MMU
  .get_unmapped_area = shmem_get_unmapped_area,
 #endif
@@ -196,7 +259,7 @@ static int newseg (key_t key, int shmflg
  if (size < SHMMIN || size > shm_ctlmax)
   return -EINVAL;

- if (shm_tot + numpages >= shm_ctlall)
+ if (shm_total + numpages >= shm_ctlall)
   return -ENOSPC;

  shp = ipc_rcu_alloc(sizeof(*shp));
@@ -249,11 +312,16 @@ static int newseg (key_t key, int shmflg
  shp->shm_file = file;
  file->f_dentry->d_inode->i_ino = shp->id;

+#ifdef CONFIG_IPC_NS
+ get_ipc_ns(current->ipc_ns);
+ file->private_data = current->ipc_ns;
+#endif
+
  /* Hugetlb ops would have already been assigned. */
  if (!(shmflg & SHM_HUGETLB))
   file->f_op = &shm_file_operations;
```

```
- shm_tot += numpages;
+ shm_total += numpages;
  shm_unlock(shp);
  return shp->id;

@@ -470,7 +538,7 @@ asmlinkage long sys_shmctl (int shmid, i
   down(&shm_ids.sem);
   shm_info.used_ids = shm_ids.in_use;
   shm_get_stat (&shm_info.shm_rss, &shm_info.shm_swp);
-  shm_info.shm_tot = shm_tot;
+  shm_info.shm_tot = shm_total;
   shm_info.swap_attempts = 0;
   shm_info.swap_successes = 0;
   err = shm_ids.max_id;
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -65,7 +65,7 @@ __initcall(ipc_init);
  * array itself.
  */

-void __init ipc_init_ids(struct ipc_ids* ids, int size)
+void __ipc_init ipc_init_ids(struct ipc_ids* ids, int size)
 {
  int i;
  sema_init(&ids->sem,1);
@@ -96,6 +96,48 @@ void __init ipc_init_ids(struct ipc_ids*
   ids->entries->p[i] = NULL;
 }

+#ifdef CONFIG_IPC_NS
+struct ipc_namespace init_ipc_ns = {
+ .cnt = ATOMIC_INIT(1),
+};
+
+struct ipc_namespace *create_ipc_ns(void)
+{
+ struct ipc_namespace *ns;
+
+ ns = kmalloc(sizeof(struct ipc_namespace), GFP_KERNEL);
+ if (ns == NULL)
+  goto out;
+
+ if (init_sem_ns(ns))
+  goto out_sem;
+ if (init_msg_ns(ns))
+  goto out_msg;
+ if (init_shm_ns(ns))
```

```
+  goto out_shm;
+
+ atomic_set(&ns->cnt, 1);
+ return ns;
+
+out_shm:
+ fini_msg_ns(ns);
+out_msg:
+ fini_sem_ns(ns);
+out_sem:
+ kfree(ns);
+out:
+ return NULL;
+}
+
+void free_ipc_ns(struct ipc_namespace *ns)
+{
+ fini_sem_ns(ns);
+ fini_msg_ns(ns);
+ fini_shm_ns(ns);
+ kfree(ns);
+}
+#endif
+
 #ifdef CONFIG_PROC_FS
 static struct file_operations sysvipc_proc_fops;
 /**
@@ -229,7 +271,9 @@ int ipc_addid(struct ipc_ids* ids, struc
  }
  return -1;
 found:
- ids->in_use++;
+ if (++ids->in_use == 1)
+  get_ipc_ns(ids->ns);
+
  if (id > ids->max_id)
   ids->max_id = id;

@@ -276,7 +320,8 @@ struct kern_ipc_perm* ipc_rmid(struct ip
  ids->entries->p[lid] = NULL;
  if(p==NULL)
   BUG();
- ids->in_use--;
+ if (--ids->in_use)
+  put_ipc_ns(ids->ns);

  if (lid == ids->max_id) {
   do {
```

```
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -8,6 +8,8 @@
 #ifndef _IPC_UTIL_H
 #define _IPC_UTIL_H

+#include <linux/config.h>
+
 #define USHRT_MAX 0xffff
 #define SEQ_MULTIPLIER (IPCMNI)

@@ -20,6 +22,7 @@ struct ipc_id_ary {
  struct kern_ipc_perm *p[0];
 };

+struct ipc_namespace;
 struct ipc_ids {
  int in_use;
  int max_id;
@@ -28,10 +31,25 @@ struct ipc_ids {
  struct semaphore sem;
  struct ipc_id_ary nullentry;
  struct ipc_id_ary* entries;
+#ifdef CONFIG_IPC_NS
+ struct ipc_namespace *ns;
+#endif
 };

+#ifdef CONFIG_IPC_NS
+int init_sem_ns(struct ipc_namespace *ns);
+int init_msg_ns(struct ipc_namespace *ns);
+int init_shm_ns(struct ipc_namespace *ns);
+void fini_sem_ns(struct ipc_namespace *ns);
+void fini_msg_ns(struct ipc_namespace *ns);
+void fini_shm_ns(struct ipc_namespace *ns);
+#define __ipc_init
+#else
+#define __ipc_init __init
+#endif
+
 struct seq_file;
-void __init ipc_init_ids(struct ipc_ids* ids, int size);
+void __ipc_init ipc_init_ids(struct ipc_ids* ids, int size);
 #ifdef CONFIG_PROC_FS
 void __init ipc_init_proc_interface(const char *path, const char *header,
         struct ipc_ids *ids,
--- a/kernel/exit.c
+++ b/kernel/exit.c
```

```
@@ -108,6 +108,7 @@ repeat:
  proc_pid_flush(proc_dentry);
  release_thread(p);
  put_uts_ns(p->uts_ns);
+ put_ipc_ns(p->ipc_ns);
  put_task_struct(p);

  p = leader;
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1193,6 +1193,7 @@ static task_t *copy_process(unsigned lon
  attach_pid(p, PIDTYPE_TGID, p->tgid);
  attach_pid(p, PIDTYPE_PID, p->pid);
  get_uts_ns(p->uts_ns);
+ get_ipc_ns(p->ipc_ns);

  nr_threads++;
  total_forks++;
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -93,6 +93,7 @@ extern char modprobe_path[];
 extern int sg_big_buff;
 #endif
 #ifdef CONFIG_SYSVIPC
+#ifndef CONFIG_IPC_NS
 extern size_t shm_ctlmax;
 extern size_t shm_ctlall;
 extern int shm_ctlmni;
@@ -100,6 +101,16 @@ extern int msg_ctlmax;
 extern int msg_ctlmnb;
 extern int msg_ctlmni;
 extern int sem_ctls[];
+#else
+#include <linux/ipc.h>
+#define shm_ctlmax (init_ipc_ns.shm_ctlmax)
+#define shm_ctlall (init_ipc_ns.shm_ctlall)
+#define shm_ctlmni (init_ipc_ns.shm_ctlmni)
+#define msg_ctlmax (init_ipc_ns.msg_ctlmax)
+#define msg_ctlmnb (init_ipc_ns.msg_ctlmnb)
+#define msg_ctlmni (init_ipc_ns.msg_ctlmni)
+#define sem_ctls (init_ipc_ns.sem_ctls)
+#endif
 #endif

 #ifdef __sparc__
```