Subject: Playing with namespaces and bloat-o-meeter
Posted by Pavel Emelianov on Wed, 26 Sep 2007 12:37:32 GMT
View Forum Message <> Reply to Message

Hi, guys!

I've noticed that compiling out all the core related to
cloning and cleaning the new namespace saves us more than
a Kbyte (!) from the vmlinux.

add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)

function	old	new	delta
copy_user_ns	-	181	+181
copy_ipcs	-	149	+149
copy_utsname	-	120	+120
shm_exit_ns	-	106	+106
sem_exit_ns	-	106	+106
msg_exit_ns	-	106	+106
freeary	-	100	+100
release_uids	-	95	+95
freeque	-	92	+92
free_nsproxy	48	99	+51
__sem_init_ns	-	45	+45
shm_init_ns	-	42	+42
sem_init_ns	-	42	+42
msg_init_ns	-	42	+42
__shm_init_ns	-	38	+38
create_new_namespaces	300	335	+35
__msg_init_ns	-	31	+31
sysvipc_proc_release	5	35	+30
free_ipc_ns	-	30	+30
do_shm_rmid	-	29	+29
shm_release	18	39	+21
free_user_ns	-	16	+16
sysvipc_proc_open	100	111	+11
do_shmat	778	787	+9
free_uts_ns	-	5	+5
sys_shmctl	1934	1907	-27
msg_init	82	47	-35
shm_init	92	47	-45
sem_init	99	44	-55
sys_msgctl	1394	1311	-83
sys_semctl	2123	2032	-91

Since there already were some questions like "do I need it
on my cellphone?" in reply to pid namespaces patches and
so on, why don't we make ALL the namespaces cloning code
under the config option to make those people happy?

Here's the proposed patch.

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

```
diff --git a/include/linux/ipc.h b/include/linux/ipc.h
index 96988d1..b882610 100644
--- a/include/linux/ipc.h
+++ b/include/linux/ipc.h
@@ -100,56 +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;
-
- 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..89f51f8
--- /dev/null
+++ b/include/linux/ipc_namespace.h
@@ -0,0 +1,67 @@
+#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;
+
+ 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
+
+#ifdef CONFIG_NS_IPC
+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;
+}
+
+static inline void put_ipc_ns(struct ipc_namespace *ns)
+{
+ kref_put(&ns->kref, free_ipc_ns);
+}
+#else
+static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
+  struct ipc_namespace *ns)
+{
+ if (flags & CLONE_NEWIPC)
+  return ERR_PTR(-EINVAL);
+
+ return ns;
+}
+
+static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
+{
+ return ns;
+}
+
+static inline void put_ipc_ns(struct ipc_namespace *ns)
+{
+}
+#endif
+#endif
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
diff --git a/include/linux/pid.h b/include/linux/pid.h
index 4817c66..ac1b47f 100644
```

```
--- a/include/linux/pid.h
+++ b/include/linux/pid.h
@@ -122,7 +122,6 @@ extern struct pid *find_ge_pid(int nr, s

 extern struct pid *alloc_pid(struct pid_namespace *ns);
 extern void FASTCALL(free_pid(struct pid *pid));
-extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

 /*
  * the helpers to get the pid's id seen from different namespaces
diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index 0135c76..1f5f915 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -6,6 +6,7 @@
 #include <linux/threads.h>
 #include <linux/nsproxy.h>
 #include <linux/kref.h>
+#include <linux/err.h>

 struct pidmap {
       atomic_t nr_free;
@@ -29,6 +30,7 @@ struct pid_namespace {

 extern struct pid_namespace init_pid_ns;

+#ifdef CONFIG_NS_PID
 static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
 {
  if (ns != &init_pid_ns)
@@ -38,12 +40,37 @@ static inline struct pid_namespace *get_

 extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *ns);
 extern void free_pid_ns(struct kref *kref);
+extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

 static inline void put_pid_ns(struct pid_namespace *ns)
 {
  if (ns != &init_pid_ns)
   kref_put(&ns->kref, free_pid_ns);
 }
+#else
+static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
+{
+ return ns;
+}
+
+static inline void put_pid_ns(struct pid_namespace *ns)
```

```
+{
+}
+
+static inline struct pid_namespace *copy_pid_ns(unsigned long flags,
+  struct pid_namespace *ns)
+{
+ if (flags & CLONE_NEWPID)
+  return ERR_PTR(-EINVAL);
+
+ return ns;
+}
+
+static inline void zap_pid_ns_processes(struct pid_namespace *ns)
+{
+ BUG();
+}
+#endif

 static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
 {
diff --git a/include/linux/sched.h b/include/linux/sched.h
diff --git a/include/linux/sem.h b/include/linux/sem.h
diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
index b5f41d4..d73080c 100644
--- a/include/linux/user_namespace.h
+++ b/include/linux/user_namespace.h
@@ -17,7 +17,7 @@ struct user_namespace {

 extern struct user_namespace init_user_ns;

-#ifdef CONFIG_USER_NS
+#ifdef CONFIG_NS_UID

 static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
 {
diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index 923db99..cea08a9 100644
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -35,6 +35,7 @@ struct new_utsname {
 #include <linux/sched.h>
 #include <linux/kref.h>
 #include <linux/nsproxy.h>
+#include <linux/err.h>
 #include <asm/atomic.h>

 struct uts_namespace {
@@ -43,6 +44,7 @@ struct uts_namespace {
```

```
 };
 extern struct uts_namespace init_uts_ns;

+#ifdef CONFIG_NS_UTS
 static inline void get_uts_ns(struct uts_namespace *ns)
 {
  kref_get(&ns->kref);
@@ -56,6 +58,25 @@ static inline void put_uts_ns(struct uts
 {
  kref_put(&ns->kref, free_uts_ns);
 }
+#else
+static inline void get_uts_ns(struct uts_namespace *ns)
+{
+}
+
+static inline void put_uts_ns(struct uts_namespace *ns)
+{
+}
+
+static inline struct uts_namespace *copy_utsname(unsigned long flags,
+     struct uts_namespace *ns)
+{
+ if (flags & CLONE_NEWUTS)
+  return ERR_PTR(-EINVAL);
+
+ return ns;
+}
+#endif
+
 static inline struct new_utsname *utsname(void)
 {
  return &current->nsproxy->uts_ns->name;
diff --git a/init/Kconfig b/init/Kconfig
index 684ccfb..ccb1575 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -206,15 +206,6 @@ config TASK_IO_ACCOUNTING

    Say N if unsure.

-config USER_NS
- bool "User Namespaces (EXPERIMENTAL)"
- default n
- depends on EXPERIMENTAL
- help
-   Support user namespaces.  This allows containers, i.e.
-   vservers, to use user namespaces to provide different
```

```
-   user info for different servers.  If unsure, say N.
-
 config AUDIT
  bool "Auditing support"
  depends on NET
@@ -369,6 +360,39 @@ config RELAY

    If unsure, say N.

+config NAMESPACES
+ bool "The namespaces support"
+ help
+   Provides the way to make tasks work with different objects using
+   the same id
+
+config NS_UTS
+ bool "Uname namespace"
+ depends on NAMESPACES
+ help
+   The utsname namespace
+
+config NS_IPC
+ bool "IPC namespace"
+ depends on NAMESPACES && SYSVIPC
+ help
+   The SYSVIPC ids namespaces
+
+config NS_PIDS
+ bool "PID namespace"
+ depends on NAMESPACES
+ help
+   Tasks see only the pids living in the same namespace and in the
+   child namespaces
+
+config NS_UID
+ bool "UID namespace"
+ depends on NAMESPACES && EXPERIMENTAL
+ help
+   Support user namespaces.  This allows containers, i.e.
+   vservers, to use user namespaces to provide different
+   user info for different servers.  If unsure, say N.
+
 config BLK_DEV_INITRD
  bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
  depends on BROKEN || !FRV
diff --git a/init/main.c b/init/main.c
diff --git a/ipc/Makefile b/ipc/Makefile
index b93bba6..b051636 100644
```

```
--- a/ipc/Makefile
+++ b/ipc/Makefile
@@ -7,4 +7,5 @@ obj-$(CONFIG_SYSVIPC) += util.o msgutil.
 obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
 obj_mq-$(CONFIG_COMPAT) += compat_mq.o
 obj-$(CONFIG_POSIX_MQUEUE) += mqueue.o msgutil.o $(obj_mq-y)
+obj-$(CONFIG_NS_IPC) += 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 b7274db..9406030 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -36,6 +36,7 @@
 #include <linux/seq_file.h>
 #include <linux/mutex.h>
 #include <linux/nsproxy.h>
+#include <linux/ipc_namespace.h>

 #include <asm/current.h>
 #include <asm/uaccess.h>
@@ -92,6 +93,7 @@ static void __msg_init_ns(struct ipc_nam
  ipc_init_ids(ids);
 }

+#ifdef CONFIG_NS_IPC
 int msg_init_ns(struct ipc_namespace *ns)
 {
  struct ipc_ids *ids;
@@ -127,6 +129,7 @@ void msg_exit_ns(struct ipc_namespace *n
  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 45c7e57..a12d52e 100644
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -82,6 +82,7 @@
 #include <linux/seq_file.h>
 #include <linux/mutex.h>
 #include <linux/nsproxy.h>
+#include <linux/ipc_namespace.h>

 #include <asm/uaccess.h>
 #include "util.h"
@@ -130,6 +131,7 @@ static void __sem_init_ns(struct ipc_nam
  ipc_init_ids(ids);
 }

+#ifdef CONFIG_NS_IPC
```

```
 int sem_init_ns(struct ipc_namespace *ns)
 {
  struct ipc_ids *ids;
@@ -165,6 +167,7 @@ void sem_exit_ns(struct ipc_namespace *n
  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 f28f2a3..07db882 100644
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -38,6 +38,7 @@
 #include <linux/mutex.h>
 #include <linux/nsproxy.h>
 #include <linux/mount.h>
+#include <linux/ipc_namespace.h>

 #include <asm/uaccess.h>

@@ -97,6 +98,7 @@ static void do_shm_rmid(struct ipc_names
   shm_destroy(ns, shp);
 }

+#ifdef CONFIG_NS_IPC
 int shm_init_ns(struct ipc_namespace *ns)
 {
  struct ipc_ids *ids;
@@ -132,6 +134,7 @@ void shm_exit_ns(struct ipc_namespace *n
  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 fd29246..44fb843 100644
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -32,6 +32,7 @@
 #include <linux/proc_fs.h>
 #include <linux/audit.h>
 #include <linux/nsproxy.h>
+#include <linux/ipc_namespace.h>
```

```
 #include <asm/unistd.h>

@@ -50,66 +51,6 @@ 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 99414a3..8972402 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/Makefile b/kernel/Makefile
index 76f782f..299261c 100644
--- a/kernel/Makefile
+++ b/kernel/Makefile
@@ -4,12 +4,11 @@

 obj-y     = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
       exit.o itimer.o time.o softirq.o resource.o \
-      sysctl.o capability.o ptrace.o timer.o user.o user_namespace.o \
+      sysctl.o capability.o ptrace.o timer.o user.o \
       signal.o sys.o kmod.o workqueue.o pid.o \
       rcupdate.o extable.o params.o posix-timers.o \
       kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
```

```
-	hrtimer.o rwsem.o latency.o nsproxy.o srcu.o \
-	utsname.o notifier.o sysctl.o
+	 hrtimer.o rwsem.o latency.o nsproxy.o srcu.o notifier.o sysctl.o

 obj-$(CONFIG_SYSCTL) += sysctl_check.o
 obj-$(CONFIG_STACKTRACE) += stacktrace.o
@@ -50,6 +49,8 @@ obj-$(CONFIG_AUDITSYSCALL) += auditsc.o
 obj-$(CONFIG_AUDIT_TREE) += audit_tree.o
 obj-$(CONFIG_KPROBES) += kprobes.o
 obj-$(CONFIG_KGDB) += kgdb.o
+obj-$(CONFIG_NS_UTS) += utsname.o
+obj-$(CONFIG_NS_UID) += user_namespace.o
 obj-$(CONFIG_SYSFS) += ksysfs.o
 obj-$(CONFIG_DETECT_SOFTLOCKUP) += softlockup.o
 obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index ee68964..5a27426 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -20,6 +20,7 @@
 #include <linux/mnt_namespace.h>
 #include <linux/utsname.h>
 #include <linux/pid_namespace.h>
+#include <linux/ipc_namespace.h>

 static struct kmem_cache *nsproxy_cachep;

diff --git a/kernel/pid.c b/kernel/pid.c
index e2e060e..a247fea 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -432,6 +432,7 @@ struct pid *find_ge_pid(int nr, struct p
 }
 EXPORT_SYMBOL_GPL(find_get_pid);

+#ifdef CONFIG_NS_PID
 struct pid_cache {
  int nr_ids;
  char name[16];
@@ -482,6 +483,11 @@ err_alloc:
  return NULL;
 }

+static __init struct kmem_cache *create_init_pid_cachep(void)
+{
+ return create_pid_cachep(1);
+}
+
```

```
 static struct pid_namespace *create_pid_namespace(int level)
 {
  struct pid_namespace *ns;
@@ -603,6 +609,13 @@ void zap_pid_ns_processes(struct pid_nam
  pid_ns->child_reaper = NULL;
  return;
 }
+#else
+static __init struct kmem_cache *create_init_pid_cachep(void)
+{
+ return kmem_cache_create("pid", sizeof(struct pid), 0,
+   SLAB_HWCACHE_ALIGN, NULL);
+}
+#endif

 /*
  * The pid hash table is scaled according to the amount of memory in the
@@ -636,7 +649,7 @@ void __init pidmap_init(void)
  set_bit(0, init_pid_ns.pidmap[0].page);
  atomic_dec(&init_pid_ns.pidmap[0].nr_free);

- init_pid_ns.pid_cachep = create_pid_cachep(1);
+ init_pid_ns.pid_cachep = create_init_pid_cachep();
  if (init_pid_ns.pid_cachep == NULL)
   panic("Can't create pid_1 cachep\n");

diff --git a/kernel/user.c b/kernel/user.c
index b45f55f..d9cac1d 100644
--- a/kernel/user.c
+++ b/kernel/user.c
@@ -17,6 +17,15 @@
 #include <linux/module.h>
 #include <linux/user_namespace.h>

+struct user_namespace init_user_ns = {
+ .kref = {
+  .refcount = ATOMIC_INIT(2),
+ },
+ .root_user = &root_user,
+};
+
+EXPORT_SYMBOL_GPL(init_user_ns);
+
 /*
  * UID task count cache, to get fast user lookup in "alloc_uid"
  * when changing user ID's (ie setuid() and friends).
@@ -199,6 +208,7 @@ void switch_uid(struct user_struct *new_
  suid_keys(current);
```

```
 }

+#ifdef CONFIG_NS_UID
 void release_uids(struct user_namespace *ns)
 {
  int i;
@@ -223,6 +233,7 @@ void release_uids(struct user_namespace

  free_uid(ns->root_user);
 }
+#endif

 static int __init uid_cache_init(void)
 {
diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
index 7af90fc..4c90062 100644
--- a/kernel/user_namespace.c
+++ b/kernel/user_namespace.c
@@ -10,17 +10,6 @@
 #include <linux/nsproxy.h>
 #include <linux/user_namespace.h>

-struct user_namespace init_user_ns = {
- .kref = {
-  .refcount = ATOMIC_INIT(2),
- },
- .root_user = &root_user,
-};
-
-EXPORT_SYMBOL_GPL(init_user_ns);
-
-#ifdef CONFIG_USER_NS
-
 /*
  * Clone a new ns copying an original user ns, setting refcount to 1
  * @old_ns: namespace to clone
@@ -84,5 +73,3 @@ void free_user_ns(struct kref *kref)
  release_uids(ns);
  kfree(ns);
 }
-
-#endif /* CONFIG_USER_NS */
diff --git a/kernel/utsname.c b/kernel/utsname.c
```

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by serue on Wed, 26 Sep 2007 13:13:49 GMT
View Forum Message <> Reply to Message

Quoting Pavel Emelyanov (xemul@openvz.org):
> Hi, guys!
>
> I've noticed that compiling out all the core related to
> cloning and cleaning the new namespace saves us more than
> a Kbyte (!) from the vmlinux.
>
> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
> function old new delta
> copy_user_ns - 181 +181
> copy_ipcs - 149 +149
> copy_utsname - 120 +120
> shm_exit_ns - 106 +106
> sem_exit_ns - 106 +106
> msg_exit_ns - 106 +106
> freeary - 100 +100
> release_uids - 95 +95
> freeque - 92 +92
> free_nsproxy 48 99 +51
> __sem_init_ns - 45 +45
> shm_init_ns - 42 +42
> sem_init_ns - 42 +42
> msg_init_ns - 42 +42
> __shm_init_ns - 38 +38
> create_new_namespaces 300 335 +35
> __msg_init_ns - 31 +31
> sysvipc_proc_release 5 35 +30
> free_ipc_ns - 30 +30
> do_shm_rmid - 29 +29
> shm_release 18 39 +21
> free_user_ns - 16 +16
> sysvipc_proc_open 100 111 +11
> do_shmat 778 787 +9
> free_uts_ns - 5 +5
> sys_shmctl 1934 1907 -27
> msg_init 82 47 -35
> shm_init 92 47 -45
> sem_init 99 44 -55
> sys_msgctl 1394 1311 -83
> sys_semctl 2123 2032 -91
>
> Since there already were some questions like "do I need it
> on my cellphone?" in reply to pid namespaces patches and
> so on, why don't we make ALL the namespaces cloning code
> under the config option to make those people happy?

>
> Here's the proposed patch.

How about a single config variable for all namespaces?

-serge

> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> ---
>
> diff --git a/include/linux/ipc.h b/include/linux/ipc.h
> index 96988d1..b882610 100644
> --- a/include/linux/ipc.h
> +++ b/include/linux/ipc.h
> @@ -100,56 +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;
> -
> - 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..89f51f8
> --- /dev/null
> +++ b/include/linux/ipc_namespace.h
> @@ -0,0 +1,67 @@
> +#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;
> +
> + 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
> +
> +#ifdef CONFIG_NS_IPC
> +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;
> +}
> +
> +static inline void put_ipc_ns(struct ipc_namespace *ns)
> +{
> + kref_put(&ns->kref, free_ipc_ns);
> +}
> +#else
> +static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
> +  struct ipc_namespace *ns)
> +{
> + if (flags & CLONE_NEWIPC)
> +  return ERR_PTR(-EINVAL);
> +
> + return ns;
> +}
> +
> +static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
> +{
> + return ns;
> +}
> +
> +static inline void put_ipc_ns(struct ipc_namespace *ns)
> +{
> +}
> +#endif
```

> +#endif
> diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
> diff --git a/include/linux/pid.h b/include/linux/pid.h
> index 4817c66..ac1b47f 100644
> --- a/include/linux/pid.h
> +++ b/include/linux/pid.h
> @@ -122,7 +122,6 @@ extern struct pid *find_ge_pid(int nr, s
>
> extern struct pid *alloc_pid(struct pid_namespace *ns);
> extern void FASTCALL(free_pid(struct pid *pid));
> -extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>
> /*
>   * the helpers to get the pid's id seen from different namespaces
> diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
> index 0135c76..1f5f915 100644
> --- a/include/linux/pid_namespace.h
> +++ b/include/linux/pid_namespace.h
> @@ -6,6 +6,7 @@
> #include <linux/threads.h>
> #include <linux/nsproxy.h>
> #include <linux/kref.h>
> +#include <linux/err.h>
>
> struct pidmap {
> 	atomic_t nr_free;
> @@ -29,6 +30,7 @@ struct pid_namespace {
>
> extern struct pid_namespace init_pid_ns;
>
> +#ifdef CONFIG_NS_PID
> static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
> {
>   if (ns != &init_pid_ns)
> @@ -38,12 +40,37 @@ static inline struct pid_namespace *get_
>
> extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *ns);
> extern void free_pid_ns(struct kref *kref);
> +extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>
> static inline void put_pid_ns(struct pid_namespace *ns)
> {
>   if (ns != &init_pid_ns)
>   kref_put(&ns->kref, free_pid_ns);
> }
> +#else
> +static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
> +{

> + return ns;
> +}
> +
> +static inline void put_pid_ns(struct pid_namespace *ns)
> +{
> +}
> +
> +static inline struct pid_namespace *copy_pid_ns(unsigned long flags,
> + struct pid_namespace *ns)
> +{
> + if (flags & CLONE_NEWPID)
> + return ERR_PTR(-EINVAL);
> +
> + return ns;
> +}
> +
> +static inline void zap_pid_ns_processes(struct pid_namespace *ns)
> +{
> + BUG();
> +}
> +#endif
>
> static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
> {
> diff --git a/include/linux/sched.h b/include/linux/sched.h
> diff --git a/include/linux/sem.h b/include/linux/sem.h
> diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
> index b5f41d4..d73080c 100644
> --- a/include/linux/user_namespace.h
> +++ b/include/linux/user_namespace.h
> @@ -17,7 +17,7 @@ struct user_namespace {
>
> extern struct user_namespace init_user_ns;
>
> -#ifdef CONFIG_USER_NS
> +#ifdef CONFIG_NS_UID
>
> static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
> {
> diff --git a/include/linux/utsname.h b/include/linux/utsname.h
> index 923db99..cea08a9 100644
> --- a/include/linux/utsname.h
> +++ b/include/linux/utsname.h
> @@ -35,6 +35,7 @@ struct new_utsname {
> #include <linux/sched.h>
> #include <linux/kref.h>
> #include <linux/nsproxy.h>
> +#include <linux/err.h>

> #include <asm/atomic.h>
>
> struct uts_namespace {
> @@ -43,6 +44,7 @@ struct uts_namespace {
> };
> extern struct uts_namespace init_uts_ns;
>
> +#ifdef CONFIG_NS_UTS
> static inline void get_uts_ns(struct uts_namespace *ns)
> {
>   kref_get(&ns->kref);
> @@ -56,6 +58,25 @@ static inline void put_uts_ns(struct uts
> {
>   kref_put(&ns->kref, free_uts_ns);
> }
> +#else
> +static inline void get_uts_ns(struct uts_namespace *ns)
> +{
> +}
> +
> +static inline void put_uts_ns(struct uts_namespace *ns)
> +{
> +}
> +
> +static inline struct uts_namespace *copy_utsname(unsigned long flags,
> +   struct uts_namespace *ns)
> +{
> + if (flags & CLONE_NEWUTS)
> + return ERR_PTR(-EINVAL);
> +
> + return ns;
> +}
> +#endif
> +
> static inline struct new_utsname *utsname(void)
> {
>   return &current->nsproxy->uts_ns->name;
> diff --git a/init/Kconfig b/init/Kconfig
> index 684ccfb..ccb1575 100644
> --- a/init/Kconfig
> +++ b/init/Kconfig
> @@ -206,15 +206,6 @@ config TASK_IO_ACCOUNTING
>
>   Say N if unsure.
>
> -config USER_NS
> - bool "User Namespaces (EXPERIMENTAL)"
> - default n

> - depends on EXPERIMENTAL
> - help
> -   Support user namespaces. This allows containers, i.e.
> -   vservers, to use user namespaces to provide different
> -   user info for different servers. If unsure, say N.
> -
> config AUDIT
>   bool "Auditing support"
>   depends on NET
> @@ -369,6 +360,39 @@ config RELAY
>
>   If unsure, say N.
>
> +config NAMESPACES
> + bool "The namespaces support"
> + help
> +   Provides the way to make tasks work with different objects using
> +   the same id
> +
> +config NS_UTS
> + bool "Uname namespace"
> + depends on NAMESPACES
> + help
> +   The utsname namespace
> +
> +config NS_IPC
> + bool "IPC namespace"
> + depends on NAMESPACES && SYSVIPC
> + help
> +   The SYSVIPC ids namespaces
> +
> +config NS_PIDS
> + bool "PID namespace"
> + depends on NAMESPACES
> + help
> +   Tasks see only the pids living in the same namespace and in the
> +   child namespaces
> +
> +config NS_UID
> + bool "UID namespace"
> + depends on NAMESPACES && EXPERIMENTAL
> + help
> +   Support user namespaces. This allows containers, i.e.
> +   vservers, to use user namespaces to provide different
> +   user info for different servers. If unsure, say N.
> +
> config BLK_DEV_INITRD
>   bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"

>   depends on BROKEN || !FRV
> diff --git a/init/main.c b/init/main.c
> diff --git a/ipc/Makefile b/ipc/Makefile
> index b93bba6..b051636 100644
> --- a/ipc/Makefile
> +++ b/ipc/Makefile
> @@ -7,4 +7,5 @@ obj-$(CONFIG_SYSVIPC) += util.o msgutil.
> obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
> obj_mq-$(CONFIG_COMPAT) += compat_mq.o
> obj-$(CONFIG_POSIX_MQUEUE) += mqueue.o msgutil.o $(obj_mq-y)
> +obj-$(CONFIG_NS_IPC) += 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 b7274db..9406030 100644
> --- a/ipc/msg.c
> +++ b/ipc/msg.c
> @@ -36,6 +36,7 @@
> #include <linux/seq_file.h>
> #include <linux/mutex.h>
> #include <linux/nsproxy.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/current.h>
> #include <asm/uaccess.h>
> @@ -92,6 +93,7 @@ static void __msg_init_ns(struct ipc_nam
>   ipc_init_ids(ids);
> }
>
> +#ifdef CONFIG_NS_IPC
> int msg_init_ns(struct ipc_namespace *ns)
> {
>   struct ipc_ids *ids;
> @@ -127,6 +129,7 @@ void msg_exit_ns(struct ipc_namespace *n
>   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 45c7e57..a12d52e 100644
> --- a/ipc/sem.c
> +++ b/ipc/sem.c
> @@ -82,6 +82,7 @@
> #include <linux/seq_file.h>
> #include <linux/mutex.h>
> #include <linux/nsproxy.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/uaccess.h>
> #include "util.h"
> @@ -130,6 +131,7 @@ static void __sem_init_ns(struct ipc_nam

>   ipc_init_ids(ids);
> }
>
> +#ifdef CONFIG_NS_IPC
> int sem_init_ns(struct ipc_namespace *ns)
> {
>   struct ipc_ids *ids;
> @@ -165,6 +167,7 @@ void sem_exit_ns(struct ipc_namespace *n
>   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 f28f2a3..07db882 100644
> --- a/ipc/shm.c
> +++ b/ipc/shm.c
> @@ -38,6 +38,7 @@
> #include <linux/mutex.h>
> #include <linux/nsproxy.h>
> #include <linux/mount.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/uaccess.h>
>
> @@ -97,6 +98,7 @@ static void do_shm_rmid(struct ipc_names
>   shm_destroy(ns, shp);
> }
>
> +#ifdef CONFIG_NS_IPC
> int shm_init_ns(struct ipc_namespace *ns)
> {
>   struct ipc_ids *ids;
> @@ -132,6 +134,7 @@ void shm_exit_ns(struct ipc_namespace *n
>   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 fd29246..44fb843 100644
> --- a/ipc/util.c
> +++ b/ipc/util.c
> @@ -32,6 +32,7 @@

> #include <linux/proc_fs.h>
> #include <linux/audit.h>
> #include <linux/nsproxy.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/unistd.h>
>
> @@ -50,66 +51,6 @@ 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 99414a3..8972402 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/Makefile b/kernel/Makefile
> index 76f782f..299261c 100644
> --- a/kernel/Makefile
> +++ b/kernel/Makefile
> @@ -4,12 +4,11 @@
>
> obj-y   = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
>   exit.o itimer.o time.o softirq.o resource.o \
> -   sysctl.o capability.o ptrace.o timer.o user.o user_namespace.o \

> +   sysctl.o capability.o ptrace.o timer.o user.o \
>   signal.o sys.o kmod.o workqueue.o pid.o \
>   rcupdate.o extable.o params.o posix-timers.o \
>   kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
> -   hrtimer.o rwsem.o latency.o nsproxy.o srcu.o \
> -   utsname.o notifier.o sysctl.o
> +   hrtimer.o rwsem.o latency.o nsproxy.o srcu.o notifier.o sysctl.o
>
> obj-$(CONFIG_SYSCTL) += sysctl_check.o
> obj-$(CONFIG_STACKTRACE) += stacktrace.o
> @@ -50,6 +49,8 @@ obj-$(CONFIG_AUDITSYSCALL) += auditsc.o
> obj-$(CONFIG_AUDIT_TREE) += audit_tree.o
> obj-$(CONFIG_KPROBES) += kprobes.o
> obj-$(CONFIG_KGDB) += kgdb.o
> +obj-$(CONFIG_NS_UTS) += utsname.o
> +obj-$(CONFIG_NS_UID) += user_namespace.o
> obj-$(CONFIG_SYSFS) += ksysfs.o
> obj-$(CONFIG_DETECT_SOFTLOCKUP) += softlockup.o
> obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
> index ee68964..5a27426 100644
> --- a/kernel/nsproxy.c
> +++ b/kernel/nsproxy.c
> @@ -20,6 +20,7 @@
> #include <linux/mnt_namespace.h>
> #include <linux/utsname.h>
> #include <linux/pid_namespace.h>
> +#include <linux/ipc_namespace.h>
>
> static struct kmem_cache *nsproxy_cachep;
>
> diff --git a/kernel/pid.c b/kernel/pid.c
> index e2e060e..a247fea 100644
> --- a/kernel/pid.c
> +++ b/kernel/pid.c
> @@ -432,6 +432,7 @@ struct pid *find_ge_pid(int nr, struct p
> }
> EXPORT_SYMBOL_GPL(find_get_pid);
>
> +#ifdef CONFIG_NS_PID
> struct pid_cache {
>   int nr_ids;
>   char name[16];
> @@ -482,6 +483,11 @@ err_alloc:
>   return NULL;
> }
>
> +static __init struct kmem_cache *create_init_pid_cachep(void)

```
> +{
> + return create_pid_cachep(1);
> +}
> +
>  static struct pid_namespace *create_pid_namespace(int level)
>  {
>   struct pid_namespace *ns;
> @@ -603,6 +609,13 @@ void zap_pid_ns_processes(struct pid_nam
>   pid_ns->child_reaper = NULL;
>   return;
>  }
> +#else
> +static __init struct kmem_cache *create_init_pid_cachep(void)
> +{
> + return kmem_cache_create("pid", sizeof(struct pid), 0,
> +   SLAB_HWCACHE_ALIGN, NULL);
> +}
> +#endif
>
>  /*
>   * The pid hash table is scaled according to the amount of memory in the
> @@ -636,7 +649,7 @@ void __init pidmap_init(void)
>   set_bit(0, init_pid_ns.pidmap[0].page);
>   atomic_dec(&init_pid_ns.pidmap[0].nr_free);
>
> - init_pid_ns.pid_cachep = create_pid_cachep(1);
> + init_pid_ns.pid_cachep = create_init_pid_cachep();
>   if (init_pid_ns.pid_cachep == NULL)
>    panic("Can't create pid_1 cachep\n");
>
> diff --git a/kernel/user.c b/kernel/user.c
> index b45f55f..d9cac1d 100644
> --- a/kernel/user.c
> +++ b/kernel/user.c
> @@ -17,6 +17,15 @@
>  #include <linux/module.h>
>  #include <linux/user_namespace.h>
>
> +struct user_namespace init_user_ns = {
> + .kref = {
> +  .refcount = ATOMIC_INIT(2),
> + },
> + .root_user = &root_user,
> +};
> +
> +EXPORT_SYMBOL_GPL(init_user_ns);
> +
>  /*
```

>   * UID task count cache, to get fast user lookup in "alloc_uid"
>   * when changing user ID's (ie setuid() and friends).
> @@ -199,6 +208,7 @@ void switch_uid(struct user_struct *new_
>   suid_keys(current);
> }
>
> +#ifdef CONFIG_NS_UID
> void release_uids(struct user_namespace *ns)
> {
>   int i;
> @@ -223,6 +233,7 @@ void release_uids(struct user_namespace
>
>   free_uid(ns->root_user);
> }
> +#endif
>
> static int __init uid_cache_init(void)
> {
> diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
> index 7af90fc..4c90062 100644
> --- a/kernel/user_namespace.c
> +++ b/kernel/user_namespace.c
> @@ -10,17 +10,6 @@
> #include <linux/nsproxy.h>
> #include <linux/user_namespace.h>
>
> -struct user_namespace init_user_ns = {
> - .kref = {
> - .refcount = ATOMIC_INIT(2),
> - },
> - .root_user = &root_user,
> -};
> -
> -EXPORT_SYMBOL_GPL(init_user_ns);
> -
> -#ifdef CONFIG_USER_NS
> -
> /*
>   * Clone a new ns copying an original user ns, setting refcount to 1
>   * @old_ns: namespace to clone
> @@ -84,5 +73,3 @@ void free_user_ns(struct kref *kref)
>   release_uids(ns);
>   kfree(ns);
> }
> -
> -#endif /* CONFIG_USER_NS */
> diff --git a/kernel/utsname.c b/kernel/utsname.c

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by Cedric Le Goater on Wed, 26 Sep 2007 13:17:39 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting Pavel Emelyanov (xemul@openvz.org):
>> Hi, guys!
>>
>> I've noticed that compiling out all the core related to
>> cloning and cleaning the new namespace saves us more than
>> a Kbyte (!) from the vmlinux.
>>
>> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
>> function old new delta
>> copy_user_ns - 181 +181
>> copy_ipcs - 149 +149
>> copy_utsname - 120 +120
>> shm_exit_ns - 106 +106
>> sem_exit_ns - 106 +106
>> msg_exit_ns - 106 +106
>> freeary - 100 +100
>> release_uids - 95 +95
>> freeque - 92 +92
>> free_nsproxy 48 99 +51
>> __sem_init_ns - 45 +45
>> shm_init_ns - 42 +42
>> sem_init_ns - 42 +42
>> msg_init_ns - 42 +42
>> __shm_init_ns - 38 +38
>> create_new_namespaces 300 335 +35
>> __msg_init_ns - 31 +31
>> sysvipc_proc_release 5 35 +30
>> free_ipc_ns - 30 +30
>> do_shm_rmid - 29 +29
>> shm_release 18 39 +21
>> free_user_ns - 16 +16
>> sysvipc_proc_open 100 111 +11
>> do_shmat 778 787 +9
>> free_uts_ns - 5 +5
>> sys_shmctl 1934 1907 -27
>> msg_init 82 47 -35
>> shm_init 92 47 -45
>> sem_init 99 44 -55

>> sys_msgctl 1394 1311 -83
>> sys_semctl 2123 2032 -91
>>
>> Since there already were some questions like "do I need it
>> on my cellphone?" in reply to pid namespaces patches and
>> so on, why don't we make ALL the namespaces cloning code
>> under the config option to make those people happy?
>>
>> Here's the proposed patch.
>
> How about a single config variable for all namespaces?

yes good idea.

C.

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

---

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by Cedric Le Goater on Wed, 26 Sep 2007 13:19:24 GMT
View Forum Message <> Reply to Message

Cedric Le Goater wrote:
> Serge E. Hallyn wrote:
>> Quoting Pavel Emelyanov (xemul@openvz.org):
>>> Hi, guys!
>>>
>>> I've noticed that compiling out all the core related to
>>> cloning and cleaning the new namespace saves us more than
>>> a Kbyte (!) from the vmlinux.
>>>
>>> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
>>> function old new delta
>>> copy_user_ns - 181 +181
>>> copy_ipcs - 149 +149
>>> copy_utsname - 120 +120
>>> shm_exit_ns - 106 +106
>>> sem_exit_ns - 106 +106
>>> msg_exit_ns - 106 +106
>>> freeary - 100 +100
>>> release_uids - 95 +95
>>> freeque - 92 +92
>>> free_nsproxy 48 99 +51
>>> __sem_init_ns - 45 +45
>>> shm_init_ns - 42 +42

```
>>> sem_init_ns                    -     42    +42
>>> msg_init_ns                    -     42    +42
>>> __shm_init_ns                   -     38    +38
>>> create_new_namespaces                 300    335    +35
>>> __msg_init_ns                  -     31    +31
>>> sysvipc_proc_release                5     35    +30
>>> free_ipc_ns                    -     30    +30
>>> do_shm_rmid                     -     29    +29
>>> shm_release                    18     39    +21
>>> free_user_ns                    -     16    +16
>>> sysvipc_proc_open                 100    111    +11
>>> do_shmat                      778    787     +9
>>> free_uts_ns                     -      5     +5
>>> sys_shmctl                   1934   1907    -27
>>> msg_init                     82     47    -35
>>> shm_init                     92     47    -45
>>> sem_init                     99     44    -55
>>> sys_msgctl                   1394   1311    -83
>>> sys_semctl                   2123   2032    -91
>>>
>>> Since there already were some questions like "do I need it
>>> on my cellphone?" in reply to pid namespaces patches and
>>> so on, why don't we make ALL the namespaces cloning code
>>> under the config option to make those people happy?
>>>
>>> Here's the proposed patch.
>> How about a single config variable for all namespaces?
>
> yes good idea.
```

oops, that done already in the patch : CONFIG_NAMESPACES

thanks :)

C.

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by Pavel Emelianov on Wed, 26 Sep 2007 13:20:06 GMT
View Forum Message <> Reply to Message

Cedric Le Goater wrote:

> Cedric Le Goater wrote:
>> Serge E. Hallyn wrote:
>>> Quoting Pavel Emelyanov (xemul@openvz.org):
>>>> Hi, guys!
>>>>
>>>> I've noticed that compiling out all the core related to
>>>> cloning and cleaning the new namespace saves us more than
>>>> a Kbyte (!) from the vmlinux.
>>>>
>>>> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
>>>> function old new delta
>>>> copy_user_ns - 181 +181
>>>> copy_ipcs - 149 +149
>>>> copy_utsname - 120 +120
>>>> shm_exit_ns - 106 +106
>>>> sem_exit_ns - 106 +106
>>>> msg_exit_ns - 106 +106
>>>> freeary - 100 +100
>>>> release_uids - 95 +95
>>>> freeque - 92 +92
>>>> free_nsproxy 48 99 +51
>>>> __sem_init_ns - 45 +45
>>>> shm_init_ns - 42 +42
>>>> sem_init_ns - 42 +42
>>>> msg_init_ns - 42 +42
>>>> __shm_init_ns - 38 +38
>>>> create_new_namespaces 300 335 +35
>>>> __msg_init_ns - 31 +31
>>>> sysvipc_proc_release 5 35 +30
>>>> free_ipc_ns - 30 +30
>>>> do_shm_rmid - 29 +29
>>>> shm_release 18 39 +21
>>>> free_user_ns - 16 +16
>>>> sysvipc_proc_open 100 111 +11
>>>> do_shmat 778 787 +9
>>>> free_uts_ns - 5 +5
>>>> sys_shmctl 1934 1907 -27
>>>> msg_init 82 47 -35
>>>> shm_init 92 47 -45
>>>> sem_init 99 44 -55
>>>> sys_msgctl 1394 1311 -83
>>>> sys_semctl 2123 2032 -91
>>>>
>>>> Since there already were some questions like "do I need it
>>>> on my cellphone?" in reply to pid namespaces patches and
>>>> so on, why don't we make ALL the namespaces cloning code
>>>> under the config option to make those people happy?
>>>>

>>>> Here's the proposed patch.
>>> How about a single config variable for all namespaces?
>> yes good idea.
>
>
> oops, that done already in the patch : CONFIG_NAMESPACES

So... Acked-by: Serge E. Hallyn and Cedric Le Goater ? :)

> thanks :)
>
> C.
>
>

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

---

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by Daniel Lezcano on Wed, 26 Sep 2007 13:23:57 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov wrote:
> Cedric Le Goater wrote:
>> Cedric Le Goater wrote:
>>> Serge E. Hallyn wrote:
>>>> Quoting Pavel Emelyanov (xemul@openvz.org):
>>>>> Hi, guys!
>>>>>
>>>>> I've noticed that compiling out all the core related to
>>>>> cloning and cleaning the new namespace saves us more than
>>>>> a Kbyte (!) from the vmlinux.
>>>>>
>>>>> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
>>>>> function old new delta
>>>>> copy_user_ns - 181 +181
>>>>> copy_ipcs - 149 +149
>>>>> copy_utsname - 120 +120
>>>>> shm_exit_ns - 106 +106
>>>>> sem_exit_ns - 106 +106
>>>>> msg_exit_ns - 106 +106
>>>>> freeary - 100 +100
>>>>> release_uids - 95 +95
>>>>> freeque - 92 +92
>>>>> free_nsproxy 48 99 +51

>>>>> __sem_init_ns - 45 +45
>>>>> shm_init_ns - 42 +42
>>>>> sem_init_ns - 42 +42
>>>>> msg_init_ns - 42 +42
>>>>> __shm_init_ns - 38 +38
>>>>> create_new_namespaces 300 335 +35
>>>>> __msg_init_ns - 31 +31
>>>>> sysvipc_proc_release 5 35 +30
>>>>> free_ipc_ns - 30 +30
>>>>> do_shm_rmid - 29 +29
>>>>> shm_release 18 39 +21
>>>>> free_user_ns - 16 +16
>>>>> sysvipc_proc_open 100 111 +11
>>>>> do_shmat 778 787 +9
>>>>> free_uts_ns - 5 +5
>>>>> sys_shmctl 1934 1907 -27
>>>>> msg_init 82 47 -35
>>>>> shm_init 92 47 -45
>>>>> sem_init 99 44 -55
>>>>> sys_msgctl 1394 1311 -83
>>>>> sys_semctl 2123 2032 -91
>>>>>
>>>>> Since there already were some questions like "do I need it
>>>>> on my cellphone?" in reply to pid namespaces patches and
>>>>> so on, why don't we make ALL the namespaces cloning code
>>>>> under the config option to make those people happy?
>>>>>
>>>>> Here's the proposed patch.
>>>> How about a single config variable for all namespaces?
>>> yes good idea.
>>
>> oops, that done already in the patch : CONFIG_NAMESPACES
>
> So... Acked-by: Serge E. Hallyn and Cedric Le Goater ? :)

That is a good idea, that will avoid to have the namespaces config all
around the menuconfig too.

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by Cedric Le Goater on Wed, 26 Sep 2007 13:24:19 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov wrote:
> Hi, guys!
>
> I've noticed that compiling out all the core related to
> cloning and cleaning the new namespace saves us more than
> a Kbyte (!) from the vmlinux.

cool.

but compared to the 5KB pid ns is adding, it's not much. I guess
anything that can be saved is good to save when you run on a cell.

> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
> function old new delta
> copy_user_ns - 181 +181
> copy_ipcs - 149 +149
> copy_utsname - 120 +120
> shm_exit_ns - 106 +106
> sem_exit_ns - 106 +106
> msg_exit_ns - 106 +106
> freeary - 100 +100
> release_uids - 95 +95
> freeque - 92 +92
> free_nsproxy 48 99 +51
> __sem_init_ns - 45 +45
> shm_init_ns - 42 +42
> sem_init_ns - 42 +42
> msg_init_ns - 42 +42
> __shm_init_ns - 38 +38
> create_new_namespaces 300 335 +35
> __msg_init_ns - 31 +31
> sysvipc_proc_release 5 35 +30
> free_ipc_ns - 30 +30
> do_shm_rmid - 29 +29
> shm_release 18 39 +21
> free_user_ns - 16 +16
> sysvipc_proc_open 100 111 +11
> do_shmat 778 787 +9
> free_uts_ns - 5 +5
> sys_shmctl 1934 1907 -27
> msg_init 82 47 -35
> shm_init 92 47 -45
> sem_init 99 44 -55
> sys_msgctl 1394 1311 -83
> sys_semctl 2123 2032 -91
>
> Since there already were some questions like "do I need it
> on my cellphone?" in reply to pid namespaces patches and

> so on, why don't we make ALL the namespaces cloning code
> under the config option to make those people happy?
>
> Here's the proposed patch.

I think I'm ok with it but it would be easier to review if you
could split it in little patchlets, at least one for each
namespace. diffstats are welcome also :)

> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> ---
>
> diff --git a/include/linux/ipc.h b/include/linux/ipc.h
> index 96988d1..b882610 100644
> --- a/include/linux/ipc.h
> +++ b/include/linux/ipc.h
> @@ -100,56 +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;
> -
> - 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..89f51f8
> --- /dev/null
> +++ b/include/linux/ipc_namespace.h

that's something i wanted to do. thanks.

> @@ -0,0 +1,67 @@
> +#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;
> +
> + 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
> +
> +#ifdef CONFIG_NS_IPC

ok so you're readding that flag. please check ipc/ipc_sysctl.c there might
be some surprises.

check the compile with CONFIG_SYSCTL=n

> +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;
> +}
> +
> +static inline void put_ipc_ns(struct ipc_namespace *ns)
> +{
> + kref_put(&ns->kref, free_ipc_ns);
> +}
> +#else
> +static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
> + struct ipc_namespace *ns)
> +{
> + if (flags & CLONE_NEWIPC)
> + return ERR_PTR(-EINVAL);
> +
> + return ns;

> +}
> +
> +static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
> +{
> + return ns;
> +}
> +
> +static inline void put_ipc_ns(struct ipc_namespace *ns)
> +{
> +}
> +#endif
> +#endif
> diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
> diff --git a/include/linux/pid.h b/include/linux/pid.h
> index 4817c66..ac1b47f 100644
> --- a/include/linux/pid.h
> +++ b/include/linux/pid.h
> @@ -122,7 +122,6 @@ extern struct pid *find_ge_pid(int nr, s
>
> extern struct pid *alloc_pid(struct pid_namespace *ns);
> extern void FASTCALL(free_pid(struct pid *pid));
> -extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>
> /*
>   * the helpers to get the pid's id seen from different namespaces
> diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
> index 0135c76..1f5f915 100644
> --- a/include/linux/pid_namespace.h
> +++ b/include/linux/pid_namespace.h
> @@ -6,6 +6,7 @@
> #include <linux/threads.h>
> #include <linux/nsproxy.h>
> #include <linux/kref.h>
> +#include <linux/err.h>
>
> struct pidmap {
> atomic_t nr_free;
> @@ -29,6 +30,7 @@ struct pid_namespace {
>
> extern struct pid_namespace init_pid_ns;
>
> +#ifdef CONFIG_NS_PID
> static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
> {
>   if (ns != &init_pid_ns)
> @@ -38,12 +40,37 @@ static inline struct pid_namespace *get_
>
> extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *ns);

```
>  extern void free_pid_ns(struct kref *kref);
> +extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>
>  static inline void put_pid_ns(struct pid_namespace *ns)
>  {
>   if (ns != &init_pid_ns)
>    kref_put(&ns->kref, free_pid_ns);
>  }
> +#else
> +static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
> +{
> + return ns;
> +}
> +
> +static inline void put_pid_ns(struct pid_namespace *ns)
> +{
> +}
> +
> +static inline struct pid_namespace *copy_pid_ns(unsigned long flags,
> +  struct pid_namespace *ns)
> +{
> + if (flags & CLONE_NEWPID)
> +  return ERR_PTR(-EINVAL);
> +
> + return ns;
> +}
> +
> +static inline void zap_pid_ns_processes(struct pid_namespace *ns)
> +{
> + BUG();
> +}
> +#endif
>
>  static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
>  {
> diff --git a/include/linux/sched.h b/include/linux/sched.h
> diff --git a/include/linux/sem.h b/include/linux/sem.h
> diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
> index b5f41d4..d73080c 100644
> --- a/include/linux/user_namespace.h
> +++ b/include/linux/user_namespace.h
> @@ -17,7 +17,7 @@ struct user_namespace {
>
>  extern struct user_namespace init_user_ns;
>
> -#ifdef CONFIG_USER_NS
> +#ifdef CONFIG_NS_UID
>
```

> static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
> {
> diff --git a/include/linux/utsname.h b/include/linux/utsname.h
> index 923db99..cea08a9 100644
> --- a/include/linux/utsname.h
> +++ b/include/linux/utsname.h
> @@ -35,6 +35,7 @@ struct new_utsname {
> #include <linux/sched.h>
> #include <linux/kref.h>
> #include <linux/nsproxy.h>
> +#include <linux/err.h>
> #include <asm/atomic.h>
>
> struct uts_namespace {
> @@ -43,6 +44,7 @@ struct uts_namespace {
> };
> extern struct uts_namespace init_uts_ns;
>
> +#ifdef CONFIG_NS_UTS
> static inline void get_uts_ns(struct uts_namespace *ns)
> {
>   kref_get(&ns->kref);
> @@ -56,6 +58,25 @@ static inline void put_uts_ns(struct uts
> {
>   kref_put(&ns->kref, free_uts_ns);
> }
> +#else
> +static inline void get_uts_ns(struct uts_namespace *ns)
> +{
> +}
> +
> +static inline void put_uts_ns(struct uts_namespace *ns)
> +{
> +}
> +
> +static inline struct uts_namespace *copy_utsname(unsigned long flags,
> +   struct uts_namespace *ns)
> +{
> + if (flags & CLONE_NEWUTS)
> + return ERR_PTR(-EINVAL);
> +
> + return ns;
> +}
> +#endif
> +
> static inline struct new_utsname *utsname(void)
> {
>   return &current->nsproxy->uts_ns->name;

> diff --git a/init/Kconfig b/init/Kconfig
> index 684ccfb..ccb1575 100644
> --- a/init/Kconfig
> +++ b/init/Kconfig
> @@ -206,15 +206,6 @@ config TASK_IO_ACCOUNTING
>
>   Say N if unsure.
>
> -config USER_NS
> - bool "User Namespaces (EXPERIMENTAL)"
> - default n
> - depends on EXPERIMENTAL
> - help
> -   Support user namespaces. This allows containers, i.e.
> -   vservers, to use user namespaces to provide different
> -   user info for different servers. If unsure, say N.
> -
> config AUDIT
>   bool "Auditing support"
>   depends on NET
> @@ -369,6 +360,39 @@ config RELAY
>
>   If unsure, say N.
>
> +config NAMESPACES
> + bool "The namespaces support"
> + help
> +   Provides the way to make tasks work with different objects using
> +   the same id
> +
> +config NS_UTS
> + bool "Uname namespace"
> + depends on NAMESPACES
> + help
> +   The utsname namespace
> +
> +config NS_IPC
> + bool "IPC namespace"
> + depends on NAMESPACES && SYSVIPC
> + help
> +   The SYSVIPC ids namespaces
> +
> +config NS_PIDS
> + bool "PID namespace"
> + depends on NAMESPACES

I'd put experimental on this one also

> + help
> +   Tasks see only the pids living in the same namespace and in the
> +   child namespaces
> +
> +config NS_UID
> + bool "UID namespace"
> + depends on NAMESPACES && EXPERIMENTAL
> + help
> +   Support user namespaces. This allows containers, i.e.
> +   vservers, to use user namespaces to provide different
> +   user info for different servers. If unsure, say N.
> +
> config BLK_DEV_INITRD
>   bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
>   depends on BROKEN || !FRV
> diff --git a/init/main.c b/init/main.c
> diff --git a/ipc/Makefile b/ipc/Makefile
> index b93bba6..b051636 100644
> --- a/ipc/Makefile
> +++ b/ipc/Makefile
> @@ -7,4 +7,5 @@ obj-$(CONFIG_SYSVIPC) += util.o msgutil.
> obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
> obj_mq-$(CONFIG_COMPAT) += compat_mq.o
> obj-$(CONFIG_POSIX_MQUEUE) += mqueue.o msgutil.o $(obj_mq-y)
> +obj-$(CONFIG_NS_IPC) += namespace.o

 ipc_namespace.c is a better name.

>
> 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 b7274db..9406030 100644
> --- a/ipc/msg.c
> +++ b/ipc/msg.c
> @@ -36,6 +36,7 @@
> #include <linux/seq_file.h>
> #include <linux/mutex.h>

> #include <linux/nsproxy.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/current.h>
> #include <asm/uaccess.h>
> @@ -92,6 +93,7 @@ static void __msg_init_ns(struct ipc_nam
>   ipc_init_ids(ids);
> }
>
> +#ifdef CONFIG_NS_IPC
> int msg_init_ns(struct ipc_namespace *ns)
> {
>   struct ipc_ids *ids;
> @@ -127,6 +129,7 @@ void msg_exit_ns(struct ipc_namespace *n
>   kfree(ns->ids[IPC_MSG_IDS]);
>   ns->ids[IPC_MSG_IDS] = NULL;
> }
> +#endif

can't this routine be in ipc_namespace.c to remove the #ifdef
CONFIG_NS_IPC ? the same apply to the other init routine.

> void __init msg_init(void)
> {
> diff --git a/ipc/namespace.c b/ipc/namespace.c

i'd call it 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 45c7e57..a12d52e 100644
> --- a/ipc/sem.c
> +++ b/ipc/sem.c
> @@ -82,6 +82,7 @@
> #include <linux/seq_file.h>
> #include <linux/mutex.h>
> #include <linux/nsproxy.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/uaccess.h>
> #include "util.h"
> @@ -130,6 +131,7 @@ static void __sem_init_ns(struct ipc_nam
> ipc_init_ids(ids);
> }
>
> +#ifdef CONFIG_NS_IPC
> int sem_init_ns(struct ipc_namespace *ns)
> {
> struct ipc_ids *ids;
> @@ -165,6 +167,7 @@ void sem_exit_ns(struct ipc_namespace *n
> 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 f28f2a3..07db882 100644
> --- a/ipc/shm.c
> +++ b/ipc/shm.c
> @@ -38,6 +38,7 @@
> #include <linux/mutex.h>
> #include <linux/nsproxy.h>
> #include <linux/mount.h>
> +#include <linux/ipc_namespace.h>
>

```
>  #include <asm/uaccess.h>
>
> @@ -97,6 +98,7 @@ static void do_shm_rmid(struct ipc_names
>    shm_destroy(ns, shp);
>  }
>
> +#ifdef CONFIG_NS_IPC
>  int shm_init_ns(struct ipc_namespace *ns)
>  {
>   struct ipc_ids *ids;
> @@ -132,6 +134,7 @@ void shm_exit_ns(struct ipc_namespace *n
>   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 fd29246..44fb843 100644
> --- a/ipc/util.c
> +++ b/ipc/util.c
> @@ -32,6 +32,7 @@
>  #include <linux/proc_fs.h>
>  #include <linux/audit.h>
>  #include <linux/nsproxy.h>
> +#include <linux/ipc_namespace.h>
>
>  #include <asm/unistd.h>
>
> @@ -50,66 +51,6 @@ 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 99414a3..8972402 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/Makefile b/kernel/Makefile
> index 76f782f..299261c 100644
> --- a/kernel/Makefile
> +++ b/kernel/Makefile
> @@ -4,12 +4,11 @@
>
>  obj-y     = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
>       exit.o itimer.o time.o softirq.o resource.o \
> -     sysctl.o capability.o ptrace.o timer.o user.o user_namespace.o \
> +     sysctl.o capability.o ptrace.o timer.o user.o \
>       signal.o sys.o kmod.o workqueue.o pid.o \
>       rcupdate.o extable.o params.o posix-timers.o \
>       kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
> -     hrtimer.o rwsem.o latency.o nsproxy.o srcu.o \
> -     utsname.o notifier.o sysctl.o
> +     hrtimer.o rwsem.o latency.o nsproxy.o srcu.o notifier.o sysctl.o
>
>  obj-$(CONFIG_SYSCTL) += sysctl_check.o
>  obj-$(CONFIG_STACKTRACE) += stacktrace.o
> @@ -50,6 +49,8 @@ obj-$(CONFIG_AUDITSYSCALL) += auditsc.o
>  obj-$(CONFIG_AUDIT_TREE) += audit_tree.o
>  obj-$(CONFIG_KPROBES) += kprobes.o
>  obj-$(CONFIG_KGDB) += kgdb.o
> +obj-$(CONFIG_NS_UTS) += utsname.o
> +obj-$(CONFIG_NS_UID) += user_namespace.o
>  obj-$(CONFIG_SYSFS) += ksysfs.o
>  obj-$(CONFIG_DETECT_SOFTLOCKUP) += softlockup.o
>  obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
> index ee68964..5a27426 100644
> --- a/kernel/nsproxy.c
> +++ b/kernel/nsproxy.c
> @@ -20,6 +20,7 @@
>  #include <linux/mnt_namespace.h>
```

> #include <linux/utsname.h>
> #include <linux/pid_namespace.h>
> +#include <linux/ipc_namespace.h>
>
> static struct kmem_cache *nsproxy_cachep;
>
> diff --git a/kernel/pid.c b/kernel/pid.c
> index e2e060e..a247fea 100644
> --- a/kernel/pid.c
> +++ b/kernel/pid.c
> @@ -432,6 +432,7 @@ struct pid *find_ge_pid(int nr, struct p
> }
> EXPORT_SYMBOL_GPL(find_get_pid);
>
> +#ifdef CONFIG_NS_PID
> struct pid_cache {
>   int nr_ids;
>   char name[16];
> @@ -482,6 +483,11 @@ err_alloc:
>   return NULL;
> }
>
> +static __init struct kmem_cache *create_init_pid_cachep(void)
> +{
> + return create_pid_cachep(1);
> +}
> +
> static struct pid_namespace *create_pid_namespace(int level)
> {
>   struct pid_namespace *ns;
> @@ -603,6 +609,13 @@ void zap_pid_ns_processes(struct pid_nam
>   pid_ns->child_reaper = NULL;
>   return;
> }
> +#else
> +static __init struct kmem_cache *create_init_pid_cachep(void)
> +{
> + return kmem_cache_create("pid", sizeof(struct pid), 0,
> +   SLAB_HWCACHE_ALIGN, NULL);
> +}
> +#endif
>
> /*
>   * The pid hash table is scaled according to the amount of memory in the
> @@ -636,7 +649,7 @@ void __init pidmap_init(void)
>   set_bit(0, init_pid_ns.pidmap[0].page);
>   atomic_dec(&init_pid_ns.pidmap[0].nr_free);
>

> - init_pid_ns.pid_cachep = create_pid_cachep(1);
> + init_pid_ns.pid_cachep = create_init_pid_cachep();
>   if (init_pid_ns.pid_cachep == NULL)
>   panic("Can't create pid_1 cachep\n");
>
> diff --git a/kernel/user.c b/kernel/user.c
> index b45f55f..d9cac1d 100644
> --- a/kernel/user.c
> +++ b/kernel/user.c
> @@ -17,6 +17,15 @@
> #include <linux/module.h>
> #include <linux/user_namespace.h>
>
> +struct user_namespace init_user_ns = {
> + .kref = {
> + .refcount = ATOMIC_INIT(2),
> + },
> + .root_user = &root_user,
> +};
> +
> +EXPORT_SYMBOL_GPL(init_user_ns);
> +
> /*
>   * UID task count cache, to get fast user lookup in "alloc_uid"
>   * when changing user ID's (ie setuid() and friends).
> @@ -199,6 +208,7 @@ void switch_uid(struct user_struct *new_
>   suid_keys(current);
> }
>
> +#ifdef CONFIG_NS_UID
> void release_uids(struct user_namespace *ns)
> {
>   int i;
> @@ -223,6 +233,7 @@ void release_uids(struct user_namespace
>
>   free_uid(ns->root_user);
> }
> +#endif
>
> static int __init uid_cache_init(void)
> {
> diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
> index 7af90fc..4c90062 100644
> --- a/kernel/user_namespace.c
> +++ b/kernel/user_namespace.c
> @@ -10,17 +10,6 @@
> #include <linux/nsproxy.h>
> #include <linux/user_namespace.h>

>
> -struct user_namespace init_user_ns = {
> - .kref = {
> - .refcount = ATOMIC_INIT(2),
> - },
> - .root_user = &root_user,
> -};
> -
> -EXPORT_SYMBOL_GPL(init_user_ns);
> -
> -#ifdef CONFIG_USER_NS
> -
> /*
> * Clone a new ns copying an original user ns, setting refcount to 1
> * @old_ns: namespace to clone
> @@ -84,5 +73,3 @@ void free_user_ns(struct kref *kref)
> release_uids(ns);
> kfree(ns);
> }
> -
> -#endif /* CONFIG_USER_NS */
> diff --git a/kernel/utsname.c b/kernel/utsname.c
>

_____

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

---

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by serue on Wed, 26 Sep 2007 13:30:19 GMT
View Forum Message <> Reply to Message

Quoting Cedric Le Goater (clg@fr.ibm.com):
> Cedric Le Goater wrote:
> > Serge E. Hallyn wrote:
> >> Quoting Pavel Emelyanov (xemul@openvz.org):
> >>> Hi, guys!
> >>>
> >>> I've noticed that compiling out all the core related to
> >>> cloning and cleaning the new namespace saves us more than
> >>> a Kbyte (!) from the vmlinux.
> >>>
> >>> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
> >>> function old new delta
> >>> copy_user_ns - 181 +181
> >>> copy_ipcs - 149 +149

> >>> copy_utsname - 120 +120
> >>> shm_exit_ns - 106 +106
> >>> sem_exit_ns - 106 +106
> >>> msg_exit_ns - 106 +106
> >>> freeary - 100 +100
> >>> release_uids - 95 +95
> >>> freeque - 92 +92
> >>> free_nsproxy 48 99 +51
> >>> __sem_init_ns - 45 +45
> >>> shm_init_ns - 42 +42
> >>> sem_init_ns - 42 +42
> >>> msg_init_ns - 42 +42
> >>> __shm_init_ns - 38 +38
> >>> create_new_namespaces 300 335 +35
> >>> __msg_init_ns - 31 +31
> >>> sysvipc_proc_release 5 35 +30
> >>> free_ipc_ns - 30 +30
> >>> do_shm_rmid - 29 +29
> >>> shm_release 18 39 +21
> >>> free_user_ns - 16 +16
> >>> sysvipc_proc_open 100 111 +11
> >>> do_shmat 778 787 +9
> >>> free_uts_ns - 5 +5
> >>> sys_shmctl 1934 1907 -27
> >>> msg_init 82 47 -35
> >>> shm_init 92 47 -45
> >>> sem_init 99 44 -55
> >>> sys_msgctl 1394 1311 -83
> >>> sys_semctl 2123 2032 -91
> >>>
> >>> Since there already were some questions like "do I need it
> >>> on my cellphone?" in reply to pid namespaces patches and
> >>> so on, why don't we make ALL the namespaces cloning code
> >>> under the config option to make those people happy?
> >>>
> >>> Here's the proposed patch.
> >> How about a single config variable for all namespaces?
> >
> > yes good idea.
>
>
> oops, that done already in the patch : CONFIG_NAMESPACES
>
> thanks :)

That at least organizes them all in Kconfig. I meant one
config variable, period.

Then instead of adding CONFIG_USER_NS and such whlie they are
experimental, put all experimental namespaces (i.e. maybe soon
user and network) under CONFIG_NAMESPACES_EXPERIMENTAL.

-serge

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

---

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by Pavel Emelianov on Wed, 26 Sep 2007 14:08:11 GMT
View Forum Message <> Reply to Message

Cedric Le Goater wrote:
> Pavel Emelyanov wrote:
>> Hi, guys!
>>
>> I've noticed that compiling out all the core related to
>> cloning and cleaning the new namespace saves us more than
>> a Kbyte (!) from the vmlinux.
>
> cool.
>
> but compared to the 5KB pid ns is adding, it's not much. I guess
> anything that can be saved is good to save when you run on a cell.

Yup, but I've already sent 3 patches to Andrew that save 1.5 KB
so we have already managed to get a half for cell users ;)

>> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
>> function old new delta
>> copy_user_ns - 181 +181
>> copy_ipcs - 149 +149
>> copy_utsname - 120 +120
>> shm_exit_ns - 106 +106
>> sem_exit_ns - 106 +106
>> msg_exit_ns - 106 +106
>> freeary - 100 +100
>> release_uids - 95 +95
>> freeque - 92 +92
>> free_nsproxy 48 99 +51
>> __sem_init_ns - 45 +45
>> shm_init_ns - 42 +42
>> sem_init_ns - 42 +42
>> msg_init_ns - 42 +42
>> __shm_init_ns - 38 +38

>> create_new_namespaces 300 335 +35
>> __msg_init_ns - 31 +31
>> sysvipc_proc_release 5 35 +30
>> free_ipc_ns - 30 +30
>> do_shm_rmid - 29 +29
>> shm_release 18 39 +21
>> free_user_ns - 16 +16
>> sysvipc_proc_open 100 111 +11
>> do_shmat 778 787 +9
>> free_uts_ns - 5 +5
>> sys_shmctl 1934 1907 -27
>> msg_init 82 47 -35
>> shm_init 92 47 -45
>> sem_init 99 44 -55
>> sys_msgctl 1394 1311 -83
>> sys_semctl 2123 2032 -91
>>
>> Since there already were some questions like "do I need it
>> on my cellphone?" in reply to pid namespaces patches and
>> so on, why don't we make ALL the namespaces cloning code
>> under the config option to make those people happy?
>>
>> Here's the proposed patch.
>
> I think I'm ok with it but it would be easier to review if you
> could split it in little patchlets, at least one for each
> namespace. diffstats are welcome also :)
>
>> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>>
>> ---
>>
>> diff --git a/include/linux/ipc.h b/include/linux/ipc.h
>> index 96988d1..b882610 100644
>> --- a/include/linux/ipc.h
>> +++ b/include/linux/ipc.h
>> @@ -100,56 +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;
>> -
>> - 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..89f51f8
```

```
>> --- /dev/null
>> +++ b/include/linux/ipc_namespace.h
>
> that's something i wanted to do. thanks.
>
>> @@ -0,0 +1,67 @@
>> +#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;
>> +
>> + 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
>> +
>> +#ifdef CONFIG_NS_IPC
>
> ok so you're readding that flag. please check ipc/ipc_sysctl.c there might
> be some surprises.
>
> check the compile with CONFIG_SYSCTL=n
>
>> +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;
>> +}
>> +
>> +static inline void put_ipc_ns(struct ipc_namespace *ns)
>> +{
>> + kref_put(&ns->kref, free_ipc_ns);
>> +}
>> +#else
>> +static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
>> +  struct ipc_namespace *ns)
>> +{
>> + if (flags & CLONE_NEWIPC)
>> +  return ERR_PTR(-EINVAL);
>> +
>> + return ns;
>> +}
>> +
>> +static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
>> +{
>> + return ns;
>> +}
>> +
>> +static inline void put_ipc_ns(struct ipc_namespace *ns)
>> +{
>> +}
>> +#endif
>> +#endif
>> diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
>> diff --git a/include/linux/pid.h b/include/linux/pid.h
>> index 4817c66..ac1b47f 100644
>> --- a/include/linux/pid.h
>> +++ b/include/linux/pid.h
>> @@ -122,7 +122,6 @@ extern struct pid *find_ge_pid(int nr, s
>>
>>  extern struct pid *alloc_pid(struct pid_namespace *ns);
>>  extern void FASTCALL(free_pid(struct pid *pid));
>> -extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>>
>>  /*
>>   * the helpers to get the pid's id seen from different namespaces
>> diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
>> index 0135c76..1f5f915 100644
>> --- a/include/linux/pid_namespace.h
>> +++ b/include/linux/pid_namespace.h
```

```
>> @@ -6,6 +6,7 @@
>>  #include <linux/threads.h>
>>  #include <linux/nsproxy.h>
>>  #include <linux/kref.h>
>> +#include <linux/err.h>
>>
>>  struct pidmap {
>>        atomic_t nr_free;
>> @@ -29,6 +30,7 @@ struct pid_namespace {
>>
>>  extern struct pid_namespace init_pid_ns;
>>
>> +#ifdef CONFIG_NS_PID
>>  static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
>>  {
>>   if (ns != &init_pid_ns)
>> @@ -38,12 +40,37 @@ static inline struct pid_namespace *get_
>>
>>  extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *ns);
>>  extern void free_pid_ns(struct kref *kref);
>> +extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>>
>>  static inline void put_pid_ns(struct pid_namespace *ns)
>>  {
>>   if (ns != &init_pid_ns)
>>    kref_put(&ns->kref, free_pid_ns);
>>  }
>> +#else
>> +static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
>> +{
>> + return ns;
>> +}
>> +
>> +static inline void put_pid_ns(struct pid_namespace *ns)
>> +{
>> +}
>> +
>> +static inline struct pid_namespace *copy_pid_ns(unsigned long flags,
>> +  struct pid_namespace *ns)
>> +{
>> + if (flags & CLONE_NEWPID)
>> +  return ERR_PTR(-EINVAL);
>> +
>> + return ns;
>> +}
>> +
>> +static inline void zap_pid_ns_processes(struct pid_namespace *ns)
>> +{
```

>> + BUG();
>> +}
>> +#endif
>>
>> static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
>> {
>> diff --git a/include/linux/sched.h b/include/linux/sched.h
>> diff --git a/include/linux/sem.h b/include/linux/sem.h
>> diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
>> index b5f41d4..d73080c 100644
>> --- a/include/linux/user_namespace.h
>> +++ b/include/linux/user_namespace.h
>> @@ -17,7 +17,7 @@ struct user_namespace {
>>
>> extern struct user_namespace init_user_ns;
>>
>> -#ifdef CONFIG_USER_NS
>> +#ifdef CONFIG_NS_UID
>>
>> static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
>> {
>> diff --git a/include/linux/utsname.h b/include/linux/utsname.h
>> index 923db99..cea08a9 100644
>> --- a/include/linux/utsname.h
>> +++ b/include/linux/utsname.h
>> @@ -35,6 +35,7 @@ struct new_utsname {
>> #include <linux/sched.h>
>> #include <linux/kref.h>
>> #include <linux/nsproxy.h>
>> +#include <linux/err.h>
>> #include <asm/atomic.h>
>>
>> struct uts_namespace {
>> @@ -43,6 +44,7 @@ struct uts_namespace {
>> };
>> extern struct uts_namespace init_uts_ns;
>>
>> +#ifdef CONFIG_NS_UTS
>> static inline void get_uts_ns(struct uts_namespace *ns)
>> {
>>   kref_get(&ns->kref);
>> @@ -56,6 +58,25 @@ static inline void put_uts_ns(struct uts
>> {
>>   kref_put(&ns->kref, free_uts_ns);
>> }
>> +#else
>> +static inline void get_uts_ns(struct uts_namespace *ns)
>> +{

```
>> +}
>> +
>> +static inline void put_uts_ns(struct uts_namespace *ns)
>> +{
>> +}
>> +
>> +static inline struct uts_namespace *copy_utsname(unsigned long flags,
>> +     struct uts_namespace *ns)
>> +{
>> + if (flags & CLONE_NEWUTS)
>> +  return ERR_PTR(-EINVAL);
>> +
>> + return ns;
>> +}
>> +#endif
>> +
>>  static inline struct new_utsname *utsname(void)
>>  {
>>   return &current->nsproxy->uts_ns->name;
>> diff --git a/init/Kconfig b/init/Kconfig
>> index 684ccfb..ccb1575 100644
>> --- a/init/Kconfig
>> +++ b/init/Kconfig
>> @@ -206,15 +206,6 @@ config TASK_IO_ACCOUNTING
>>
>>     Say N if unsure.
>>
>> -config USER_NS
>> - bool "User Namespaces (EXPERIMENTAL)"
>> - default n
>> - depends on EXPERIMENTAL
>> - help
>> -   Support user namespaces.  This allows containers, i.e.
>> -   vservers, to use user namespaces to provide different
>> -   user info for different servers.  If unsure, say N.
>> -
>>  config AUDIT
>>   bool "Auditing support"
>>   depends on NET
>> @@ -369,6 +360,39 @@ config RELAY
>>
>>     If unsure, say N.
>>
>> +config NAMESPACES
>> + bool "The namespaces support"
>> + help
>> +   Provides the way to make tasks work with different objects using
>> +   the same id
```

>> +
>> +config NS_UTS
>> + bool "Uname namespace"
>> + depends on NAMESPACES
>> + help
>> +   The utsname namespace
>> +
>> +config NS_IPC
>> + bool "IPC namespace"
>> + depends on NAMESPACES && SYSVIPC
>> + help
>> +   The SYSVIPC ids namespaces
>> +
>> +config NS_PIDS
>> + bool "PID namespace"
>> + depends on NAMESPACES
>
> I'd put experimental on this one also
>
>> + help
>> +   Tasks see only the pids living in the same namespace and in the
>> +   child namespaces
>> +
>> +config NS_UID
>> + bool "UID namespace"
>> + depends on NAMESPACES && EXPERIMENTAL
>> + help
>> +   Support user namespaces. This allows containers, i.e.
>> +   vservers, to use user namespaces to provide different
>> +   user info for different servers. If unsure, say N.
>> +
>> config BLK_DEV_INITRD
>>   bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
>>   depends on BROKEN || !FRV
>> diff --git a/init/main.c b/init/main.c
>> diff --git a/ipc/Makefile b/ipc/Makefile
>> index b93bba6..b051636 100644
>> --- a/ipc/Makefile
>> +++ b/ipc/Makefile
>> @@ -7,4 +7,5 @@ obj-$(CONFIG_SYSVIPC) += util.o msgutil.
>> obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
>> obj_mq-$(CONFIG_COMPAT) += compat_mq.o
>> obj-$(CONFIG_POSIX_MQUEUE) += mqueue.o msgutil.o $(obj_mq-y)
>> +obj-$(CONFIG_NS_IPC) += namespace.o
>
> ipc_namespace.c is a better name.
>
>> 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 b7274db..9406030 100644
>> --- a/ipc/msg.c
>> +++ b/ipc/msg.c
>> @@ -36,6 +36,7 @@
>> #include <linux/seq_file.h>
>> #include <linux/mutex.h>
>> #include <linux/nsproxy.h>
>> +#include <linux/ipc_namespace.h>
>>
>> #include <asm/current.h>
>> #include <asm/uaccess.h>
>> @@ -92,6 +93,7 @@ static void __msg_init_ns(struct ipc_nam
>>   ipc_init_ids(ids);
>> }
>>
>> +#ifdef CONFIG_NS_IPC
>> int msg_init_ns(struct ipc_namespace *ns)
>> {
>>   struct ipc_ids *ids;
>> @@ -127,6 +129,7 @@ void msg_exit_ns(struct ipc_namespace *n
>>   kfree(ns->ids[IPC_MSG_IDS]);
>>   ns->ids[IPC_MSG_IDS] = NULL;
>> }
>> +#endif
>
> can't this routine be in ipc_namespace.c to remove the #ifdef
> CONFIG_NS_IPC ? the same apply to the other init routine.
>
>> void __init msg_init(void)
>> {
>> diff --git a/ipc/namespace.c b/ipc/namespace.c
>
> i'd call it 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 45c7e57..a12d52e 100644
>> --- a/ipc/sem.c
>> +++ b/ipc/sem.c
>> @@ -82,6 +82,7 @@
>>  #include <linux/seq_file.h>
>>  #include <linux/mutex.h>
>>  #include <linux/nsproxy.h>
>> +#include <linux/ipc_namespace.h>
>>
>>  #include <asm/uaccess.h>
>>  #include "util.h"
>> @@ -130,6 +131,7 @@ static void __sem_init_ns(struct ipc_nam
>>   ipc_init_ids(ids);
>>  }
>>
>> +#ifdef CONFIG_NS_IPC
>>  int sem_init_ns(struct ipc_namespace *ns)
>>  {
>>   struct ipc_ids *ids;
```

```
>> @@ -165,6 +167,7 @@ void sem_exit_ns(struct ipc_namespace *n
>>   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 f28f2a3..07db882 100644
>> --- a/ipc/shm.c
>> +++ b/ipc/shm.c
>> @@ -38,6 +38,7 @@
>>  #include <linux/mutex.h>
>>  #include <linux/nsproxy.h>
>>  #include <linux/mount.h>
>> +#include <linux/ipc_namespace.h>
>>
>>  #include <asm/uaccess.h>
>>
>> @@ -97,6 +98,7 @@ static void do_shm_rmid(struct ipc_names
>>    shm_destroy(ns, shp);
>>  }
>>
>> +#ifdef CONFIG_NS_IPC
>>  int shm_init_ns(struct ipc_namespace *ns)
>>  {
>>   struct ipc_ids *ids;
>> @@ -132,6 +134,7 @@ void shm_exit_ns(struct ipc_namespace *n
>>   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 fd29246..44fb843 100644
>> --- a/ipc/util.c
>> +++ b/ipc/util.c
>> @@ -32,6 +32,7 @@
>>  #include <linux/proc_fs.h>
>>  #include <linux/audit.h>
>>  #include <linux/nsproxy.h>
>> +#include <linux/ipc_namespace.h>
>>
>>  #include <asm/unistd.h>
>>
```

```
>> @@ -50,66 +51,6 @@ 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 99414a3..8972402 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/Makefile b/kernel/Makefile
>> index 76f782f..299261c 100644
>> --- a/kernel/Makefile
>> +++ b/kernel/Makefile
>> @@ -4,12 +4,11 @@
>>
>>  obj-y     = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
>>       exit.o itimer.o time.o softirq.o resource.o \
>> -     sysctl.o capability.o ptrace.o timer.o user.o user_namespace.o \
>> +     sysctl.o capability.o ptrace.o timer.o user.o \
>>       signal.o sys.o kmod.o workqueue.o pid.o \
>>       rcupdate.o extable.o params.o posix-timers.o \
>>       kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
>> -     hrtimer.o rwsem.o latency.o nsproxy.o srcu.o \
>> -     utsname.o notifier.o sysctl.o
>> +     hrtimer.o rwsem.o latency.o nsproxy.o srcu.o notifier.o sysctl.o
```

```
>>
>>  obj-$(CONFIG_SYSCTL) += sysctl_check.o
>>  obj-$(CONFIG_STACKTRACE) += stacktrace.o
>> @@ -50,6 +49,8 @@ obj-$(CONFIG_AUDITSYSCALL) += auditsc.o
>>  obj-$(CONFIG_AUDIT_TREE) += audit_tree.o
>>  obj-$(CONFIG_KPROBES) += kprobes.o
>>  obj-$(CONFIG_KGDB) += kgdb.o
>> +obj-$(CONFIG_NS_UTS) += utsname.o
>> +obj-$(CONFIG_NS_UID) += user_namespace.o
>>  obj-$(CONFIG_SYSFS) += ksysfs.o
>>  obj-$(CONFIG_DETECT_SOFTLOCKUP) += softlockup.o
>>  obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
>> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
>> index ee68964..5a27426 100644
>> --- a/kernel/nsproxy.c
>> +++ b/kernel/nsproxy.c
>> @@ -20,6 +20,7 @@
>>  #include <linux/mnt_namespace.h>
>>  #include <linux/utsname.h>
>>  #include <linux/pid_namespace.h>
>> +#include <linux/ipc_namespace.h>
>>
>>  static struct kmem_cache *nsproxy_cachep;
>>
>> diff --git a/kernel/pid.c b/kernel/pid.c
>> index e2e060e..a247fea 100644
>> --- a/kernel/pid.c
>> +++ b/kernel/pid.c
>> @@ -432,6 +432,7 @@ struct pid *find_ge_pid(int nr, struct p
>>  }
>>  EXPORT_SYMBOL_GPL(find_get_pid);
>>
>> +#ifdef CONFIG_NS_PID
>>  struct pid_cache {
>>   int nr_ids;
>>   char name[16];
>> @@ -482,6 +483,11 @@ err_alloc:
>>   return NULL;
>>  }
>>
>> +static __init struct kmem_cache *create_init_pid_cachep(void)
>> +{
>> + return create_pid_cachep(1);
>> +}
>> +
>>  static struct pid_namespace *create_pid_namespace(int level)
>>  {
>>   struct pid_namespace *ns;
```

```
>> @@ -603,6 +609,13 @@ void zap_pid_ns_processes(struct pid_nam
>>   pid_ns->child_reaper = NULL;
>>   return;
>>  }
>> +#else
>> +static __init struct kmem_cache *create_init_pid_cachep(void)
>> +{
>> + return kmem_cache_create("pid", sizeof(struct pid), 0,
>> +   SLAB_HWCACHE_ALIGN, NULL);
>> +}
>> +#endif
>>
>>  /*
>>   * The pid hash table is scaled according to the amount of memory in the
>> @@ -636,7 +649,7 @@ void __init pidmap_init(void)
>>   set_bit(0, init_pid_ns.pidmap[0].page);
>>   atomic_dec(&init_pid_ns.pidmap[0].nr_free);
>>
>> - init_pid_ns.pid_cachep = create_pid_cachep(1);
>> + init_pid_ns.pid_cachep = create_init_pid_cachep();
>>   if (init_pid_ns.pid_cachep == NULL)
>>    panic("Can't create pid_1 cachep\n");
>>
>> diff --git a/kernel/user.c b/kernel/user.c
>> index b45f55f..d9cac1d 100644
>> --- a/kernel/user.c
>> +++ b/kernel/user.c
>> @@ -17,6 +17,15 @@
>>  #include <linux/module.h>
>>  #include <linux/user_namespace.h>
>>
>> +struct user_namespace init_user_ns = {
>> + .kref = {
>> +  .refcount = ATOMIC_INIT(2),
>> + },
>> + .root_user = &root_user,
>> +};
>> +
>> +EXPORT_SYMBOL_GPL(init_user_ns);
>> +
>>  /*
>>   * UID task count cache, to get fast user lookup in "alloc_uid"
>>   * when changing user ID's (ie setuid() and friends).
>> @@ -199,6 +208,7 @@ void switch_uid(struct user_struct *new_
>>   suid_keys(current);
>>  }
>>
>> +#ifdef CONFIG_NS_UID
```

```
>>  void release_uids(struct user_namespace *ns)
>>  {
>>   int i;
>> @@ -223,6 +233,7 @@ void release_uids(struct user_namespace
>>
>>   free_uid(ns->root_user);
>>  }
>> +#endif
>>
>>  static int __init uid_cache_init(void)
>>  {
>> diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
>> index 7af90fc..4c90062 100644
>> --- a/kernel/user_namespace.c
>> +++ b/kernel/user_namespace.c
>> @@ -10,17 +10,6 @@
>>  #include <linux/nsproxy.h>
>>  #include <linux/user_namespace.h>
>>
>> -struct user_namespace init_user_ns = {
>> - .kref = {
>> -  .refcount = ATOMIC_INIT(2),
>> - },
>> - .root_user = &root_user,
>> -};
>> -
>> -EXPORT_SYMBOL_GPL(init_user_ns);
>> -
>> -#ifdef CONFIG_USER_NS
>> -
>>  /*
>>   * Clone a new ns copying an original user ns, setting refcount to 1
>>   * @old_ns: namespace to clone
>> @@ -84,5 +73,3 @@ void free_user_ns(struct kref *kref)
>>   release_uids(ns);
>>   kfree(ns);
>>  }
>> -
>> -#endif /* CONFIG_USER_NS */
>> diff --git a/kernel/utsname.c b/kernel/utsname.c
>>
>
>
```

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by Pavel Emelianov on Wed, 26 Sep 2007 14:20:51 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov wrote:
> Hi, guys!
>
> I've noticed that compiling out all the core related to
> cloning and cleaning the new namespace saves us more than
> a Kbyte (!) from the vmlinux.

Sorry guys, this patch is a bit broken :P

CONFIG_NS_PID is used in kernel/pid.c while the real
option is CONFIG_NS_PIDS :(

I've remade the patch and it actually saves 2KB :) I
will send it to Andrew soon.

add/remove: 27/0 grow/shrink: 11/7 up/down: 2477/-340 (2137)

function	old	new	delta
copy_pid_ns	-	537	+537
copy_user_ns	-	181	+181
copy_ipcs	-	149	+149
zap_pid_ns_processes	-	130	+130
copy_utsname	-	120	+120
shm_exit_ns	-	106	+106
sem_exit_ns	-	106	+106
msg_exit_ns	-	106	+106
freeary	-	100	+100
release_uids	-	95	+95
freeque	-	92	+92
free_nsproxy	48	123	+75
create_new_namespaces	300	358	+58
free_pid_ns	-	56	+56
pid_namespaces_init	-	48	+48
__sem_init_ns	-	45	+45
shm_init_ns	-	42	+42
sem_init_ns	-	42	+42
msg_init_ns	-	42	+42
__shm_init_ns	-	38	+38
__msg_init_ns	-	31	+31
sysvipc_proc_release	5	35	+30
proc_kill_sb	5	35	+30
free_ipc_ns	-	30	+30
do_shm_rmid	-	29	+29
proc_set_super	13	38	+25
shm_release	18	39	+21
put_pid	75	95	+20

```
alloc_pid                          687    706    +19
pid_caches_mutex                     -     16    +16
free_user_ns                         -     16    +16
sysvipc_proc_open                  100    111    +11
do_shmat                           778    787     +9
pid_caches_lh                        -      8     +8
free_uts_ns                          -      5     +5
pid_ns_cachep                        -      4     +4
__initcall_pid_namespaces_init6      -      4     +4
do_exit                         1855   1856     +1
show_stat                       1665   1661     -4
sys_shmctl                      1934   1907    -27
msg_init                          82     47    -35
shm_init                          92     47    -45
sem_init                          99     44    -55
sys_msgctl                      1394   1311    -83
sys_semctl                      2123   2032    -91


> add/remove: 19/0 grow/shrink: 6/6 up/down: 1532/-336 (1196)
> function                          old     new   delta
> copy_user_ns                        -     181    +181
> copy_ipcs                           -     149    +149
> copy_utsname                        -     120    +120
> shm_exit_ns                         -     106    +106
> sem_exit_ns                         -     106    +106
> msg_exit_ns                         -     106    +106
> freeary                           -     100    +100
> release_uids                        -      95     +95
> freeque                           -      92     +92
> free_nsproxy                       48      99     +51
> __sem_init_ns                       -      45     +45
> shm_init_ns                         -      42     +42
> sem_init_ns                         -      42     +42
> msg_init_ns                         -      42     +42
> __shm_init_ns                       -      38     +38
> create_new_namespaces             300     335     +35
> __msg_init_ns                       -      31     +31
> sysvipc_proc_release                5      35     +30
> free_ipc_ns                         -      30     +30
> do_shm_rmid                         -      29     +29
> shm_release                        18      39     +21
> free_user_ns                        -      16     +16
> sysvipc_proc_open                 100     111     +11
> do_shmat                          778     787      +9
> free_uts_ns                         -      5      +5
> sys_shmctl                       1934    1907     -27
> msg_init                           82      47     -35
```

> shm_init 92 47 -45
> sem_init 99 44 -55
> sys_msgctl 1394 1311 -83
> sys_semctl 2123 2032 -91
>
> Since there already were some questions like "do I need it
> on my cellphone?" in reply to pid namespaces patches and
> so on, why don't we make ALL the namespaces cloning code
> under the config option to make those people happy?
>
> Here's the proposed patch.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> ---
>
> diff --git a/include/linux/ipc.h b/include/linux/ipc.h
> index 96988d1..b882610 100644
> --- a/include/linux/ipc.h
> +++ b/include/linux/ipc.h
> @@ -100,56 +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;
> -
> - 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..89f51f8
> --- /dev/null
> +++ b/include/linux/ipc_namespace.h
> @@ -0,0 +1,67 @@
> +#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;
> +
> + 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
> +
> +#ifdef CONFIG_NS_IPC
> +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;
> +}
> +
> +static inline void put_ipc_ns(struct ipc_namespace *ns)
> +{
> + kref_put(&ns->kref, free_ipc_ns);
> +}
> +#else
> +static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
> +  struct ipc_namespace *ns)
> +{
> + if (flags & CLONE_NEWIPC)
> +  return ERR_PTR(-EINVAL);
> +
> + return ns;
> +}
> +
> +static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
> +{
> + return ns;
> +}
```

> +
> +static inline void put_ipc_ns(struct ipc_namespace *ns)
> +{
> +}
> +#endif
> +#endif
> diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
> diff --git a/include/linux/pid.h b/include/linux/pid.h
> index 4817c66..ac1b47f 100644
> --- a/include/linux/pid.h
> +++ b/include/linux/pid.h
> @@ -122,7 +122,6 @@ extern struct pid *find_ge_pid(int nr, s
>
> extern struct pid *alloc_pid(struct pid_namespace *ns);
> extern void FASTCALL(free_pid(struct pid *pid));
> -extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>
> /*
>   * the helpers to get the pid's id seen from different namespaces
> diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
> index 0135c76..1f5f915 100644
> --- a/include/linux/pid_namespace.h
> +++ b/include/linux/pid_namespace.h
> @@ -6,6 +6,7 @@
> #include <linux/threads.h>
> #include <linux/nsproxy.h>
> #include <linux/kref.h>
> +#include <linux/err.h>
>
> struct pidmap {
> 	atomic_t nr_free;
> @@ -29,6 +30,7 @@ struct pid_namespace {
>
> extern struct pid_namespace init_pid_ns;
>
> +#ifdef CONFIG_NS_PID
> static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
> {
>   if (ns != &init_pid_ns)
> @@ -38,12 +40,37 @@ static inline struct pid_namespace *get_
>
> extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *ns);
> extern void free_pid_ns(struct kref *kref);
> +extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>
> static inline void put_pid_ns(struct pid_namespace *ns)
> {
>   if (ns != &init_pid_ns)

>   kref_put(&ns->kref, free_pid_ns);
> }
> +#else
> +static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
> +{
> + return ns;
> +}
> +
> +static inline void put_pid_ns(struct pid_namespace *ns)
> +{
> +}
> +
> +static inline struct pid_namespace *copy_pid_ns(unsigned long flags,
> + struct pid_namespace *ns)
> +{
> + if (flags & CLONE_NEWPID)
> + return ERR_PTR(-EINVAL);
> +
> + return ns;
> +}
> +
> +static inline void zap_pid_ns_processes(struct pid_namespace *ns)
> +{
> + BUG();
> +}
> +#endif
>
> static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)
> {
> diff --git a/include/linux/sched.h b/include/linux/sched.h
> diff --git a/include/linux/sem.h b/include/linux/sem.h
> diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
> index b5f41d4..d73080c 100644
> --- a/include/linux/user_namespace.h
> +++ b/include/linux/user_namespace.h
> @@ -17,7 +17,7 @@ struct user_namespace {
>
> extern struct user_namespace init_user_ns;
>
> -#ifdef CONFIG_USER_NS
> +#ifdef CONFIG_NS_UID
>
> static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
> {
> diff --git a/include/linux/utsname.h b/include/linux/utsname.h
> index 923db99..cea08a9 100644
> --- a/include/linux/utsname.h
> +++ b/include/linux/utsname.h

```
> @@ -35,6 +35,7 @@ struct new_utsname {
>  #include <linux/sched.h>
>  #include <linux/kref.h>
>  #include <linux/nsproxy.h>
> +#include <linux/err.h>
>  #include <asm/atomic.h>
>
>  struct uts_namespace {
> @@ -43,6 +44,7 @@ struct uts_namespace {
>  };
>  extern struct uts_namespace init_uts_ns;
>
> +#ifdef CONFIG_NS_UTS
>  static inline void get_uts_ns(struct uts_namespace *ns)
>  {
>   kref_get(&ns->kref);
> @@ -56,6 +58,25 @@ static inline void put_uts_ns(struct uts
>  {
>   kref_put(&ns->kref, free_uts_ns);
>  }
> +#else
> +static inline void get_uts_ns(struct uts_namespace *ns)
> +{
> +}
> +
> +static inline void put_uts_ns(struct uts_namespace *ns)
> +{
> +}
> +
> +static inline struct uts_namespace *copy_utsname(unsigned long flags,
> +     struct uts_namespace *ns)
> +{
> + if (flags & CLONE_NEWUTS)
> +  return ERR_PTR(-EINVAL);
> +
> + return ns;
> +}
> +#endif
> +
>  static inline struct new_utsname *utsname(void)
>  {
>   return &current->nsproxy->uts_ns->name;
> diff --git a/init/Kconfig b/init/Kconfig
> index 684ccfb..ccb1575 100644
> --- a/init/Kconfig
> +++ b/init/Kconfig
> @@ -206,15 +206,6 @@ config TASK_IO_ACCOUNTING
>
```

>   Say N if unsure.
>
> -config USER_NS
> - bool "User Namespaces (EXPERIMENTAL)"
> - default n
> - depends on EXPERIMENTAL
> - help
> -   Support user namespaces. This allows containers, i.e.
> -   vservers, to use user namespaces to provide different
> -   user info for different servers. If unsure, say N.
> -
> config AUDIT
>   bool "Auditing support"
>   depends on NET
> @@ -369,6 +360,39 @@ config RELAY
>
>   If unsure, say N.
>
> +config NAMESPACES
> + bool "The namespaces support"
> + help
> +   Provides the way to make tasks work with different objects using
> +   the same id
> +
> +config NS_UTS
> + bool "Uname namespace"
> + depends on NAMESPACES
> + help
> +   The utsname namespace
> +
> +config NS_IPC
> + bool "IPC namespace"
> + depends on NAMESPACES && SYSVIPC
> + help
> +   The SYSVIPC ids namespaces
> +
> +config NS_PIDS
> + bool "PID namespace"
> + depends on NAMESPACES
> + help
> +   Tasks see only the pids living in the same namespace and in the
> +   child namespaces
> +
> +config NS_UID
> + bool "UID namespace"
> + depends on NAMESPACES && EXPERIMENTAL
> + help
> +   Support user namespaces. This allows containers, i.e.

> +   vservers, to use user namespaces to provide different
> +   user info for different servers. If unsure, say N.
> +
> config BLK_DEV_INITRD
>   bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
>   depends on BROKEN || !FRV
> diff --git a/init/main.c b/init/main.c
> diff --git a/ipc/Makefile b/ipc/Makefile
> index b93bba6..b051636 100644
> --- a/ipc/Makefile
> +++ b/ipc/Makefile
> @@ -7,4 +7,5 @@ obj-$(CONFIG_SYSVIPC) += util.o msgutil.
> obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
> obj_mq-$(CONFIG_COMPAT) += compat_mq.o
> obj-$(CONFIG_POSIX_MQUEUE) += mqueue.o msgutil.o $(obj_mq-y)
> +obj-$(CONFIG_NS_IPC) += 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 b7274db..9406030 100644
> --- a/ipc/msg.c
> +++ b/ipc/msg.c
> @@ -36,6 +36,7 @@
> #include <linux/seq_file.h>
> #include <linux/mutex.h>
> #include <linux/nsproxy.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/current.h>
> #include <asm/uaccess.h>
> @@ -92,6 +93,7 @@ static void __msg_init_ns(struct ipc_nam
>   ipc_init_ids(ids);
> }
>
> +#ifdef CONFIG_NS_IPC
> int msg_init_ns(struct ipc_namespace *ns)
> {

>   struct ipc_ids *ids;
> @@ -127,6 +129,7 @@ void msg_exit_ns(struct ipc_namespace *n
>   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 45c7e57..a12d52e 100644
> --- a/ipc/sem.c
> +++ b/ipc/sem.c
> @@ -82,6 +82,7 @@
>  #include <linux/seq_file.h>
>  #include <linux/mutex.h>
>  #include <linux/nsproxy.h>
```

> +#include <linux/ipc_namespace.h>
>
> #include <asm/uaccess.h>
> #include "util.h"
> @@ -130,6 +131,7 @@ static void __sem_init_ns(struct ipc_nam
>   ipc_init_ids(ids);
> }
>
> +#ifdef CONFIG_NS_IPC
> int sem_init_ns(struct ipc_namespace *ns)
> {
>   struct ipc_ids *ids;
> @@ -165,6 +167,7 @@ void sem_exit_ns(struct ipc_namespace *n
>   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 f28f2a3..07db882 100644
> --- a/ipc/shm.c
> +++ b/ipc/shm.c
> @@ -38,6 +38,7 @@
> #include <linux/mutex.h>
> #include <linux/nsproxy.h>
> #include <linux/mount.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/uaccess.h>
>
> @@ -97,6 +98,7 @@ static void do_shm_rmid(struct ipc_names
>   shm_destroy(ns, shp);
> }
>
> +#ifdef CONFIG_NS_IPC
> int shm_init_ns(struct ipc_namespace *ns)
> {
>   struct ipc_ids *ids;
> @@ -132,6 +134,7 @@ void shm_exit_ns(struct ipc_namespace *n
>   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 fd29246..44fb843 100644
> --- a/ipc/util.c
> +++ b/ipc/util.c
> @@ -32,6 +32,7 @@
> #include <linux/proc_fs.h>
> #include <linux/audit.h>
> #include <linux/nsproxy.h>
> +#include <linux/ipc_namespace.h>
>
> #include <asm/unistd.h>
>
> @@ -50,66 +51,6 @@ 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 99414a3..8972402 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/Makefile b/kernel/Makefile
> index 76f782f..299261c 100644
> --- a/kernel/Makefile
> +++ b/kernel/Makefile

```
> @@ -4,12 +4,11 @@
>
>  obj-y     = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
>        exit.o itimer.o time.o softirq.o resource.o \
> -     sysctl.o capability.o ptrace.o timer.o user.o user_namespace.o \
> +     sysctl.o capability.o ptrace.o timer.o user.o \
>        signal.o sys.o kmod.o workqueue.o pid.o \
>        rcupdate.o extable.o params.o posix-timers.o \
>        kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
> -     hrtimer.o rwsem.o latency.o nsproxy.o srcu.o \
> -     utsname.o notifier.o sysctl.o
> +     hrtimer.o rwsem.o latency.o nsproxy.o srcu.o notifier.o sysctl.o
>
>  obj-$(CONFIG_SYSCTL) += sysctl_check.o
>  obj-$(CONFIG_STACKTRACE) += stacktrace.o
> @@ -50,6 +49,8 @@ obj-$(CONFIG_AUDITSYSCALL) += auditsc.o
>  obj-$(CONFIG_AUDIT_TREE) += audit_tree.o
>  obj-$(CONFIG_KPROBES) += kprobes.o
>  obj-$(CONFIG_KGDB) += kgdb.o
> +obj-$(CONFIG_NS_UTS) += utsname.o
> +obj-$(CONFIG_NS_UID) += user_namespace.o
>  obj-$(CONFIG_SYSFS) += ksysfs.o
>  obj-$(CONFIG_DETECT_SOFTLOCKUP) += softlockup.o
>  obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
> index ee68964..5a27426 100644
> --- a/kernel/nsproxy.c
> +++ b/kernel/nsproxy.c
> @@ -20,6 +20,7 @@
>  #include <linux/mnt_namespace.h>
>  #include <linux/utsname.h>
>  #include <linux/pid_namespace.h>
> +#include <linux/ipc_namespace.h>
>
>  static struct kmem_cache *nsproxy_cachep;
>
> diff --git a/kernel/pid.c b/kernel/pid.c
> index e2e060e..a247fea 100644
> --- a/kernel/pid.c
> +++ b/kernel/pid.c
> @@ -432,6 +432,7 @@ struct pid *find_ge_pid(int nr, struct p
>  }
>  EXPORT_SYMBOL_GPL(find_get_pid);
>
> +#ifdef CONFIG_NS_PID
>  struct pid_cache {
>   int nr_ids;
>   char name[16];
```

```
> @@ -482,6 +483,11 @@ err_alloc:
>   return NULL;
>  }
>
> +static __init struct kmem_cache *create_init_pid_cachep(void)
> +{
> + return create_pid_cachep(1);
> +}
> +
>  static struct pid_namespace *create_pid_namespace(int level)
>  {
>   struct pid_namespace *ns;
> @@ -603,6 +609,13 @@ void zap_pid_ns_processes(struct pid_nam
>   pid_ns->child_reaper = NULL;
>   return;
>  }
> +#else
> +static __init struct kmem_cache *create_init_pid_cachep(void)
> +{
> + return kmem_cache_create("pid", sizeof(struct pid), 0,
> +    SLAB_HWCACHE_ALIGN, NULL);
> +}
> +#endif
>
>  /*
>   * The pid hash table is scaled according to the amount of memory in the
> @@ -636,7 +649,7 @@ void __init pidmap_init(void)
>   set_bit(0, init_pid_ns.pidmap[0].page);
>   atomic_dec(&init_pid_ns.pidmap[0].nr_free);
>
> - init_pid_ns.pid_cachep = create_pid_cachep(1);
> + init_pid_ns.pid_cachep = create_init_pid_cachep();
>   if (init_pid_ns.pid_cachep == NULL)
>    panic("Can't create pid_1 cachep\n");
>
> diff --git a/kernel/user.c b/kernel/user.c
> index b45f55f..d9cac1d 100644
> --- a/kernel/user.c
> +++ b/kernel/user.c
> @@ -17,6 +17,15 @@
>  #include <linux/module.h>
>  #include <linux/user_namespace.h>
>
> +struct user_namespace init_user_ns = {
> + .kref = {
> +  .refcount = ATOMIC_INIT(2),
> + },
> + .root_user = &root_user,
```

> +};
> +
> +EXPORT_SYMBOL_GPL(init_user_ns);
> +
> /*
>   * UID task count cache, to get fast user lookup in "alloc_uid"
>   * when changing user ID's (ie setuid() and friends).
> @@ -199,6 +208,7 @@ void switch_uid(struct user_struct *new_
>   suid_keys(current);
> }
>
> +#ifdef CONFIG_NS_UID
> void release_uids(struct user_namespace *ns)
> {
>   int i;
> @@ -223,6 +233,7 @@ void release_uids(struct user_namespace
>
>   free_uid(ns->root_user);
> }
> +#endif
>
> static int __init uid_cache_init(void)
> {
> diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
> index 7af90fc..4c90062 100644
> --- a/kernel/user_namespace.c
> +++ b/kernel/user_namespace.c
> @@ -10,17 +10,6 @@
> #include <linux/nsproxy.h>
> #include <linux/user_namespace.h>
>
> -struct user_namespace init_user_ns = {
> - .kref = {
> - .refcount = ATOMIC_INIT(2),
> - },
> - .root_user = &root_user,
> -};
> -
> -EXPORT_SYMBOL_GPL(init_user_ns);
> -
> -#ifdef CONFIG_USER_NS
> -
> /*
>   * Clone a new ns copying an original user ns, setting refcount to 1
>   * @old_ns: namespace to clone
> @@ -84,5 +73,3 @@ void free_user_ns(struct kref *kref)
>   release_uids(ns);
>   kfree(ns);

> }
> -
> -#endif /* CONFIG_USER_NS */
> diff --git a/kernel/utsname.c b/kernel/utsname.c
>

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

---

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by ebiederm on Wed, 26 Sep 2007 14:45:03 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov <xemul@openvz.org> writes:

>>
>> +config NAMESPACES
>> + bool "The namespaces support"
>> + help
>> +   Provides the way to make tasks work with different objects using
>> +   the same id
>> +
>> +config NS_UTS
>> + bool "Uname namespace"
>> + depends on NAMESPACES
>> + help
>> +   The utsname namespace
>> +
>> +config NS_IPC
>> + bool "IPC namespace"
>> + depends on NAMESPACES && SYSVIPC
>> + help
>> +   The SYSVIPC ids namespaces
>> +
>> +config NS_PIDS
>> + bool "PID namespace"
>> + depends on NAMESPACES
>> + help
>> +   Tasks see only the pids living in the same namespace and in the
>> +   child namespaces
>> +
>> +config NS_UID
>> + bool "UID namespace"
>> + depends on NAMESPACES && EXPERIMENTAL
>> + help

>> +   Support user namespaces. This allows containers, i.e.
>> +   vservers, to use user namespaces to provide different
>> +   user info for different servers. If unsure, say N.
>> +
>> config BLK_DEV_INITRD
>>   bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
>>   depends on BROKEN || !FRV

The reason we removed these options earlier was a maintenance issue
and the fact we could not actually compile out the namespaces.

If we don't cause maintenance complications I think the general
idea is fine. But please. This all should show up under
CONFIG_EMBEDDED since the only purpose is to save space.

While things are experimental there is an additional purpose of
not exposing people to broken or partially working code, so it
does make sense to have an option there.

Eric

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by Pavel Emelianov on Wed, 26 Sep 2007 14:49:42 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Pavel Emelyanov <xemul@openvz.org> writes:
>
>>>
>>> +config NAMESPACES
>>> + bool "The namespaces support"
>>> + help
>>> +   Provides the way to make tasks work with different objects using
>>> +   the same id
>>> +
>>> +config NS_UTS
>>> + bool "Uname namespace"
>>> + depends on NAMESPACES
>>> + help
>>> +   The utsname namespace
>>> +
>>> +config NS_IPC

>>> + bool "IPC namespace"
>>> + depends on NAMESPACES && SYSVIPC
>>> + help
>>> +   The SYSVIPC ids namespaces
>>> +
>>> +config NS_PIDS
>>> + bool "PID namespace"
>>> + depends on NAMESPACES
>>> + help
>>> +   Tasks see only the pids living in the same namespace and in the
>>> +   child namespaces
>>> +
>>> +config NS_UID
>>> + bool "UID namespace"
>>> + depends on NAMESPACES && EXPERIMENTAL
>>> + help
>>> +   Support user namespaces. This allows containers, i.e.
>>> +   vservers, to use user namespaces to provide different
>>> +   user info for different servers. If unsure, say N.
>>> +
>>> config BLK_DEV_INITRD
>>>   bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
>>>   depends on BROKEN || !FRV
>
> The reason we removed these options earlier was a maintenance issue
> and the fact we could not actually compile out the namespaces.

I do not propose to compile out the namespaces, I just propose
to compile out the code that does the clone and release of new
namespaces. This is absolutely painless.

> If we don't cause maintenance complications I think the general
> idea is fine. But please. This all should show up under
> CONFIG_EMBEDDED since the only purpose is to save space.

Hm... Ok, but I also try to save the vmlinux size on my home
PC, so I'd be happy if I could just throw these things out.

Anyway - I will move the CONFIG_NAMESPACES to be selectable
with the EMBEDDED only.

> While things are experimental there is an additional purpose of
> not exposing people to broken or partially working code, so it
> does make sense to have an option there.

Ok, thanks.

So your accolades, can they be transformed into Acked-by-s or

just mentioned in the patch like "reviewed and approved by ..."?

> Eric
>

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

---

Subject: Re: Playing with namespaces and bloat-o-meeter
Posted by serue on Wed, 26 Sep 2007 15:42:36 GMT
View Forum Message <> Reply to Message

Quoting Pavel Emelyanov (xemul@openvz.org):
> Eric W. Biederman wrote:
> > Pavel Emelyanov <xemul@openvz.org> writes:
> >
> >>>
> >>> +config NAMESPACES
> >>> + bool "The namespaces support"
> >>> + help
> >>> + Provides the way to make tasks work with different objects using
> >>> + the same id
> >>> +
> >>> +config NS_UTS
> >>> + bool "Uname namespace"
> >>> + depends on NAMESPACES
> >>> + help
> >>> + The utsname namespace
> >>> +
> >>> +config NS_IPC
> >>> + bool "IPC namespace"
> >>> + depends on NAMESPACES && SYSVIPC
> >>> + help
> >>> + The SYSVIPC ids namespaces
> >>> +
> >>> +config NS_PIDS
> >>> + bool "PID namespace"
> >>> + depends on NAMESPACES
> >>> + help
> >>> + Tasks see only the pids living in the same namespace and in the
> >>> + child namespaces
> >>> +
> >>> +config NS_UID
> >>> + bool "UID namespace"
> >>> + depends on NAMESPACES && EXPERIMENTAL

> >>> + help
> >>> +   Support user namespaces. This allows containers, i.e.
> >>> +   vservers, to use user namespaces to provide different
> >>> +   user info for different servers. If unsure, say N.
> >>> +
> >>> config BLK_DEV_INITRD
> >>>   bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
> >>>   depends on BROKEN || !FRV
> >
> > The reason we removed these options earlier was a maintenance issue
> > and the fact we could not actually compile out the namespaces.
>
> I do not propose to compile out the namespaces, I just propose
> to compile out the code that does the clone and release of new
> namespaces. This is absolutely painless.
>
> > If we don't cause maintenance complications I think the general
> > idea is fine. But please. This all should show up under
> > CONFIG_EMBEDDED since the only purpose is to save space.
>
> Hm... Ok, but I also try to save the vmlinux size on my home
> PC, so I'd be happy if I could just throw these things out.
>
> Anyway - I will move the CONFIG_NAMESPACES to be selectable
> with the EMBEDDED only.
>
> > While things are experimental there is an additional purpose of
> > not exposing people to broken or partially working code, so it
> > does make sense to have an option there.
>
> Ok, thanks.
>
> So your accolades, can they be transformed into Acked-by-s or
> just mentioned in the patch like "reviewed and approved by ..."?

I for one have no objection to the idea itself. There have been several
good suggestions though so I would like to see one more round here to
which I can add an Acked-by.

thanks,
-serge