Subject: [PATCH 4/6] Cleanup the code managed with the USER_NS option
Posted by Pavel Emelianov on Wed, 14 Nov 2007 11:36:57 GMT
View Forum Message <> Reply to Message

Make the user_namespace.o compilation depend on this option and
move the init_user_ns into user.c file to make the kernel compile
and work without the namespaces support. This make the user
namespace code be organized similar to other namespaces'.

Also mask the USER_NS option as "depend on NAMESPACES".

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

```
diff --git a/init/Kconfig b/init/Kconfig
index fe73fe6..825f10c 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 PID_NS
  bool "PID Namespaces (EXPERIMENTAL)"
  default n
@@ -427,6 +418,14 @@ config IPC_NS
    In this namespace tasks work with IPC ids which correspond to
    different IPC objects in different namespaces

+config USER_NS
+ bool "User namespace (EXPERIMENTAL)"
+ depends on NAMESPACES && EXPERIMENTAL
+ help
+   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/kernel/Makefile b/kernel/Makefile
index 19d2411..d01cb7b 100644
--- a/kernel/Makefile
+++ b/kernel/Makefile
@@ -4,7 +4,7 @@

 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 \
@@ -49,6 +49,7 @@ obj-$(CONFIG_AUDITSYSCALL) += auditsc.o
 obj-$(CONFIG_AUDIT_TREE) += audit_tree.o
 obj-$(CONFIG_KPROBES) += kprobes.o
 obj-$(CONFIG_UTS_NS) += utsname.o
+obj-$(CONFIG_USER_NS) += user_namespace.o
 obj-$(CONFIG_SYSFS) += ksysfs.o
 obj-$(CONFIG_DETECT_SOFTLOCKUP) += softlockup.o
 obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
diff --git a/kernel/user.c b/kernel/user.c
index 74a1ddf..624e3d7 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).
@@ -430,6 +439,7 @@ void switch_uid(struct user_struct *new_user)
  suid_keys(current);
 }

+#ifdef CONFIG_USER_NS
```

```
 void release_uids(struct user_namespace *ns)
 {
  int i;
@@ -454,6 +464,7 @@ void release_uids(struct user_namespace *ns)

  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 */
--
1.5.3.4
```

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