
Subject: [PATCH 1/9] namespaces: add nsproxy
Posted by [serue](#) on Thu, 18 May 2006 15:48:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch adds a nsproxy structure to the task struct. Later patches will move the fs namespace pointer into this structure, and introduce a new utsname namespace into the nsproxy.

The vserver and openvz functionality, then, would be implemented in large part by virtualizing/isolating more and more resources into namespaces, each contained in the nsproxy.

Signed-off-by: Serge Hallyn <serue@us.ibm.com>

```
arch/alpha/kernel/init_task.c | 2 +
arch/arm/kernel/init_task.c   | 2 +
arch/arm26/kernel/init_task.c | 2 +
arch/frv/kernel/init_task.c   | 2 +
arch/h8300/kernel/init_task.c | 2 +
arch/i386/kernel/init_task.c   | 2 +
arch/ia64/kernel/init_task.c   | 2 +
arch/m32r/kernel/init_task.c   | 2 +
arch/m68knommu/kernel/init_task.c | 2 +
arch/mips/kernel/init_task.c   | 2 +
arch/parisc/kernel/init_task.c | 2 +
arch/powerpc/kernel/init_task.c | 2 +
arch/s390/kernel/init_task.c   | 2 +
arch/sh/kernel/init_task.c     | 2 +
arch/sh64/kernel/init_task.c   | 2 +
arch/sparc/kernel/init_task.c   | 2 +
arch/sparc64/kernel/init_task.c | 2 +
arch/um/kernel/init_task.c     | 2 +
arch/v850/kernel/init_task.c   | 2 +
arch/x86_64/kernel/init_task.c | 2 +
include/linux/init_task.h      | 7 +++
include/linux/nsproxy.h        | 45 +++++
include/linux/sched.h          | 2 +
kernel/Makefile                | 2 -
kernel/exit.c                  | 7 +++
kernel/fork.c                   | 18 ++++++-
kernel/nsproxy.c                | 78 +++++
27 files changed, 197 insertions(+), 2 deletions(-)
create mode 100644 include/linux/nsproxy.h
create mode 100644 kernel/nsproxy.c
```

7c65cea6d1931f03867fad978f33072a4b0a4602

```
diff --git a/arch/alpha/kernel/init_task.c b/arch/alpha/kernel/init_task.c
index 835d09a..83d0902 100644
--- a/arch/alpha/kernel/init_task.c
+++ b/arch/alpha/kernel/init_task.c
@@ -5,6 +5,7 @@
#include <linux/init_task.h>
#include <linux/fs.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>
#include <asm/uaccess.h>
```

```
@@ -13,6 +14,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
struct task_struct init_task = INIT_TASK(init_task);
```

```
EXPORT_SYMBOL(init_mm);
diff --git a/arch/arm/kernel/init_task.c b/arch/arm/kernel/init_task.c
index a00cca0..80f5eeb 100644
--- a/arch/arm/kernel/init_task.c
+++ b/arch/arm/kernel/init_task.c
@@ -8,6 +8,7 @@
#include <linux/init.h>
#include <linux/init_task.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>
```

```
#include <asm/uaccess.h>
#include <asm/pgtable.h>
@@ -17,6 +18,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);

EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/arm26/kernel/init_task.c b/arch/arm26/kernel/init_task.c
index 4191565..678c7b5 100644
--- a/arch/arm26/kernel/init_task.c
+++ b/arch/arm26/kernel/init_task.c
@@ -11,6 +11,7 @@
#include <linux/init.h>
#include <linux/init_task.h>
#include <linux/mqueue.h>
```

```
+#include <linux/nsproxy.h>
```

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

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

```
@@ -20,6 +21,7 @@ static struct files_struct init_files =  
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);  
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);  
struct mm_struct init_mm = INIT_MM(init_mm);  
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/frv/kernel/init_task.c b/arch/frv/kernel/init_task.c
```

```
index 2299393..5ec2742 100644
```

```
--- a/arch/frv/kernel/init_task.c
```

```
+++ b/arch/frv/kernel/init_task.c
```

```
@@ -5,6 +5,7 @@
```

```
#include <linux/init_task.h>
```

```
#include <linux/fs.h>
```

```
#include <linux/mqueue.h>
```

```
+#include <linux/nsproxy.h>
```

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

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

```
@@ -15,6 +16,7 @@ static struct files_struct init_files =  
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);  
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);  
struct mm_struct init_mm = INIT_MM(init_mm);  
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/h8300/kernel/init_task.c b/arch/h8300/kernel/init_task.c
```

```
index 19272c2..ef5755a 100644
```

```
--- a/arch/h8300/kernel/init_task.c
```

```
+++ b/arch/h8300/kernel/init_task.c
```

```
@@ -8,6 +8,7 @@
```

```
#include <linux/init_task.h>
```

```
#include <linux/fs.h>
```

```
#include <linux/mqueue.h>
```

```
+#include <linux/nsproxy.h>
```

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

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

```
@@ -17,6 +18,7 @@ static struct files_struct init_files =  
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);  
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);  
struct mm_struct init_mm = INIT_MM(init_mm);
```

```
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/i386/kernel/init_task.c b/arch/i386/kernel/init_task.c
```

```
index cff95d1..bd97f69 100644
```

```
--- a/arch/i386/kernel/init_task.c
```

```
+++ b/arch/i386/kernel/init_task.c
```

```
@@ -5,6 +5,7 @@
```

```
#include <linux/init_task.h>
```

```
#include <linux/fs.h>
```

```
#include <linux/mqueue.h>
```

```
+#include <linux/nsproxy.h>
```

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

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

```
@@ -15,6 +16,7 @@ static struct files_struct init_files =
```

```
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
```

```
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
```

```
struct mm_struct init_mm = INIT_MM(init_mm);
```

```
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/ia64/kernel/init_task.c b/arch/ia64/kernel/init_task.c
```

```
index b69c397..2d62471 100644
```

```
--- a/arch/ia64/kernel/init_task.c
```

```
+++ b/arch/ia64/kernel/init_task.c
```

```
@@ -12,6 +12,7 @@
```

```
#include <linux/sched.h>
```

```
#include <linux/init_task.h>
```

```
#include <linux/mqueue.h>
```

```
+#include <linux/nsproxy.h>
```

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

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

```
@@ -21,6 +22,7 @@ static struct files_struct init_files =
```

```
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
```

```
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
```

```
struct mm_struct init_mm = INIT_MM(init_mm);
```

```
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/m32r/kernel/init_task.c b/arch/m32r/kernel/init_task.c
```

```
index 9e508fd..0057475 100644
```

```
--- a/arch/m32r/kernel/init_task.c
```

```
+++ b/arch/m32r/kernel/init_task.c
```

```
@@ -7,6 +7,7 @@
#include <linux/init_task.h>
#include <linux/fs.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>
```

```
#include <asm/uaccess.h>
#include <asm/pgtable.h>
@@ -16,6 +17,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/m68knommu/kernel/init_task.c b/arch/m68knommu/kernel/init_task.c
index 3897043..b99fc6d 100644
```

```
--- a/arch/m68knommu/kernel/init_task.c
+++ b/arch/m68knommu/kernel/init_task.c
```

```
@@ -8,6 +8,7 @@
#include <linux/init_task.h>
#include <linux/fs.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>
```

```
#include <asm/uaccess.h>
#include <asm/pgtable.h>
@@ -17,6 +18,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/mips/kernel/init_task.c b/arch/mips/kernel/init_task.c
index aeda7f5..dfe47e6 100644
```

```
--- a/arch/mips/kernel/init_task.c
+++ b/arch/mips/kernel/init_task.c
```

```
@@ -4,6 +4,7 @@
#include <linux/init_task.h>
#include <linux/fs.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>
```

```
#include <asm/thread_info.h>
#include <asm/uaccess.h>
```

```
@@ -14,6 +15,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/parisc/kernel/init_task.c b/arch/parisc/kernel/init_task.c
```

```
index 8384bf9..c0c43e2 100644
```

```
--- a/arch/parisc/kernel/init_task.c
```

```
+++ b/arch/parisc/kernel/init_task.c
```

```
@@ -28,6 +28,7 @@
```

```
#include <linux/init.h>
```

```
#include <linux/init_task.h>
```

```
#include <linux/mqueue.h>
```

```
+#include <linux/nsproxy.h>
```

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

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

```
@@ -38,6 +39,7 @@ static struct files_struct init_files =
```

```
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
```

```
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
```

```
struct mm_struct init_mm = INIT_MM(init_mm);
```

```
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/powerpc/kernel/init_task.c b/arch/powerpc/kernel/init_task.c
```

```
index 941043a..e24ace6 100644
```

```
--- a/arch/powerpc/kernel/init_task.c
```

```
+++ b/arch/powerpc/kernel/init_task.c
```

```
@@ -5,6 +5,7 @@
```

```
#include <linux/init_task.h>
```

```
#include <linux/fs.h>
```

```
#include <linux/mqueue.h>
```

```
+#include <linux/nsproxy.h>
```

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

```
static struct fs_struct init_fs = INIT_FS;
```

```
@@ -12,6 +13,7 @@ static struct files_struct init_files =
```

```
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
```

```
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
```

```
struct mm_struct init_mm = INIT_MM(init_mm);
```

```
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```

diff --git a/arch/s390/kernel/init_task.c b/arch/s390/kernel/init_task.c
index d73a740..0918921 100644
--- a/arch/s390/kernel/init_task.c
+++ b/arch/s390/kernel/init_task.c
@@ -11,6 +11,7 @@
#include <linux/sched.h>
#include <linux/init_task.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>

#include <asm/uaccess.h>
#include <asm/pgtable.h>
@@ -20,6 +21,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);

EXPORT_SYMBOL(init_mm);

```

```

diff --git a/arch/sh/kernel/init_task.c b/arch/sh/kernel/init_task.c
index 44053ea..81caf0f 100644
--- a/arch/sh/kernel/init_task.c
+++ b/arch/sh/kernel/init_task.c
@@ -3,6 +3,7 @@
#include <linux/sched.h>
#include <linux/init_task.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>

#include <asm/uaccess.h>
#include <asm/pgtable.h>
@@ -12,6 +13,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);

EXPORT_SYMBOL(init_mm);

```

```

diff --git a/arch/sh64/kernel/init_task.c b/arch/sh64/kernel/init_task.c
index de2d07d..0c95f40 100644
--- a/arch/sh64/kernel/init_task.c
+++ b/arch/sh64/kernel/init_task.c
@@ -14,6 +14,7 @@
#include <linux/sched.h>
#include <linux/init_task.h>
#include <linux/mqueue.h>

```

```

+#include <linux/nsproxy.h>

#include <asm/uaccess.h>
#include <asm/pgtable.h>
@@ -23,6 +24,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);

struct pt_regs fake_swapper_regs;

diff --git a/arch/sparc/kernel/init_task.c b/arch/sparc/kernel/init_task.c
index fc31de6..a73926d 100644
--- a/arch/sparc/kernel/init_task.c
+++ b/arch/sparc/kernel/init_task.c
@@ -3,6 +3,7 @@
#include <linux/sched.h>
#include <linux/init_task.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>

#include <asm/pgtable.h>
#include <asm/uaccess.h>
@@ -12,6 +13,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
struct task_struct init_task = INIT_TASK(init_task);

EXPORT_SYMBOL(init_mm);
diff --git a/arch/sparc64/kernel/init_task.c b/arch/sparc64/kernel/init_task.c
index 329b38f..f1e9a4b 100644
--- a/arch/sparc64/kernel/init_task.c
+++ b/arch/sparc64/kernel/init_task.c
@@ -3,6 +3,7 @@
#include <linux/sched.h>
#include <linux/init_task.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>

#include <asm/pgtable.h>
#include <asm/uaccess.h>
@@ -13,6 +14,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);

```



```
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/um/kernel/init_task.c b/arch/um/kernel/init_task.c
```

```
index 49ed5dd..11188af 100644
```

```
--- a/arch/um/kernel/init_task.c
```

```
+++ b/arch/um/kernel/init_task.c
```

```
@@ -9,6 +9,7 @@
```

```
#include "linux/sched.h"
```

```
#include "linux/init_task.h"
```

```
#include "linux/mqueue.h"
```

```
+#include "linux/nsproxy.h"
```

```
#include "asm/uaccess.h"
```

```
#include "asm/pgtable.h"
```

```
#include "user_util.h"
```

```
@@ -17,6 +18,7 @@
```

```
static struct fs_struct init_fs = INIT_FS;
```

```
struct mm_struct init_mm = INIT_MM(init_mm);
```

```
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
static struct files_struct init_files = INIT_FILES;
```

```
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
```

```
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
```

```
diff --git a/arch/v850/kernel/init_task.c b/arch/v850/kernel/init_task.c
```

```
index ed2f93c..9d2de75 100644
```

```
--- a/arch/v850/kernel/init_task.c
```

```
+++ b/arch/v850/kernel/init_task.c
```

```
@@ -16,6 +16,7 @@
```

```
#include <linux/init_task.h>
```

```
#include <linux/fs.h>
```

```
#include <linux/mqueue.h>
```

```
+#include <linux/nsproxy.h>
```

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

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

```
@@ -25,6 +26,7 @@ static struct files_struct init_files =
```

```
static struct signal_struct init_signals = INIT_SIGNALS (init_signals);
```

```
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
```

```
struct mm_struct init_mm = INIT_MM (init_mm);
```

```
+struct nsproxy init_nsproxy = INIT_NS_PROXY(init_nsproxy);
```

```
EXPORT_SYMBOL(init_mm);
```

```
diff --git a/arch/x86_64/kernel/init_task.c b/arch/x86_64/kernel/init_task.c
```

```
index ce31d90..1c87ea0 100644
```

```
--- a/arch/x86_64/kernel/init_task.c
```

```
+++ b/arch/x86_64/kernel/init_task.c
```

```

@@ -5,6 +5,7 @@
#include <linux/init_task.h>
#include <linux/fs.h>
#include <linux/mqueue.h>
+#include <linux/nsproxy.h>

#include <asm/uaccess.h>
#include <asm/pgtable.h>
@@ -15,6 +16,7 @@ static struct files_struct init_files =
static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
struct mm_struct init_mm = INIT_MM(init_mm);
+struct nsproxy init_nsproxy = INIT_NSProxy(init_nsproxy);

EXPORT_SYMBOL(init_mm);

diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index 41ecbb8..79ec4ea 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -66,6 +66,12 @@
.session = 1, \
}

+extern struct nsproxy init_nsproxy;
+#define INIT_NSProxy(nsproxy) { \
+ .count = ATOMIC_INIT(1), \
+ .nslock = SPIN_LOCK_UNLOCKED, \
+}
+
#define INIT_SIGHAND(sighand) { \
.count = ATOMIC_INIT(1), \
.action = { { { .sa_handler = NULL, } }, }, \
@@ -114,6 +120,7 @@ extern struct group_info init_groups;
.files = &init_files, \
.signal = &init_signals, \
.sighand = &init_sighand, \
+ .nsproxy = &init_nsproxy, \
.pending = { \
.list = LIST_HEAD_INIT(tsk.pending.list), \
.signal = {{0}}}, \
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
new file mode 100644
index 0000000..7bdebfa
--- /dev/null
+++ b/include/linux/nsproxy.h
@@ -0,0 +1,45 @@
+#ifndef _LINUX_NSProxy_H

```

```

+#define _LINUX_NSProxy_H
+
+#include <linux/spinlock.h>
+#include <linux/sched.h>
+
+/*
+ * A structure to contain pointers to all per-process
+ * namespaces - fs (mount), uts, network, sysvipc, etc.
+ *
+ * 'count' is the number of tasks holding a reference.
+ * The count for each namespace, then, will be the number
+ * of nsproxies pointing to it, not the number of tasks.
+ *
+ * The nsproxy is shared by tasks which share all namespaces.
+ * As soon as a single namespace is cloned or unshared, the
+ * nsproxy is copied.
+ */
+struct nsproxy {
+ atomic_t count;
+ spinlock_t nslock;
+};
+extern struct nsproxy init_nsproxy;
+
+struct nsproxy *dup_namespaces(struct nsproxy *orig);
+int copy_namespaces(int flags, struct task_struct *tsk);
+void get_task_namespaces(struct task_struct *tsk);
+void free_nsproxy(struct nsproxy *ns);
+
+static inline void put_nsproxy(struct nsproxy *ns)
+{
+ if (atomic_dec_and_test(&ns->count)) {
+ free_nsproxy(ns);
+ }
+}
+
+static inline void exit_task_namespaces(struct task_struct *p)
+{
+ struct nsproxy *ns = p->nsproxy;
+ if (ns) {
+ put_nsproxy(ns);
+ p->nsproxy = NULL;
+ }
+}
+#endif
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 29b7d4f..4c0bbb3 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h

```

```
@@ -236,6 +236,7 @@ extern signed long schedule_timeout_unin
asmlinkage void schedule(void);
```

```
struct namespace;
+struct nsproxy;
```

```
/* Maximum number of active map areas.. This is a random (large) number */
#define DEFAULT_MAX_MAP_COUNT 65536
```

```
@@ -808,6 +809,7 @@ struct task_struct {
```

```
    struct files_struct *files;
```

```
/* namespace */
```

```
    struct namespace *namespace;
```

```
+ struct nsproxy *nsproxy;
```

```
/* signal handlers */
```

```
    struct signal_struct *signal;
```

```
    struct sighand_struct *sighand;
```

```
diff --git a/kernel/Makefile b/kernel/Makefile
```

```
index 58908f9..215fb33 100644
```

```
--- a/kernel/Makefile
```

```
+++ b/kernel/Makefile
```

```
@@ -8,7 +8,7 @@ obj-y    = sched.o fork.o exec_domain.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
```

```
+    hrtimer.o nsproxy.o
```

```
obj-$(CONFIG_DEBUG_MUTEXES) += mutex-debug.o
```

```
obj-$(CONFIG_FUTEX) += futex.o
```

```
diff --git a/kernel/exit.c b/kernel/exit.c
```

```
index e95b932..da2fc84 100644
```

```
--- a/kernel/exit.c
```

```
+++ b/kernel/exit.c
```

```
@@ -36,6 +36,7 @@
```

```
#include <linux/compat.h>
```

```
#include <linux/pipe_fs_i.h>
```

```
#include <linux/audit.h> /* for audit_free() */
```

```
+#include <linux/nsproxy.h>
```

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

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

```
@@ -413,9 +414,14 @@ void daemonize(const char *name, ...)
```

```
    fs = init_task.fs;
```

```
    current->fs = fs;
```

```
    atomic_inc(&fs->count);
```

```
+
```

```
    exit_namespace(current);
```

```
+ exit_task_namespaces(current);
```

```

    current->namespace = init_task.namespace;
+ current->nsproxy = init_task.nsproxy;
    get_namespace(current->namespace);
+ get_task_namespaces(current);
+
    exit_files(current);
    current->files = init_task.files;
    atomic_inc(&current->files->count);
@@ -919,6 +925,7 @@ fastcall NORET_TYPE void do_exit(long co
    __exit_files(tsk);
    __exit_fs(tsk);
    exit_namespace(tsk);
+ exit_task_namespaces(tsk);
    exit_thread();
    cpuset_exit(tsk);
    exit_keys(tsk);
diff --git a/kernel/fork.c b/kernel/fork.c
index ac8100e..60303c3 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -44,6 +44,7 @@
#include <linux/rmap.h>
#include <linux/acct.h>
#include <linux/cn_proc.h>
+#include <linux/nsproxy.h>

#include <asm/pgtable.h>
#include <asm/pgalloc.h>
@@ -1060,8 +1061,10 @@ static task_t *copy_process(unsigned lon
    goto bad_fork_cleanup_signal;
    if ((retval = copy_keys(clone_flags, p)))
        goto bad_fork_cleanup_mm;
- if ((retval = copy_namespace(clone_flags, p)))
+ if ((retval = copy_namespaces(clone_flags, p)))
    goto bad_fork_cleanup_keys;
+ if ((retval = copy_namespace(clone_flags, p)))
+ goto bad_fork_cleanup_namespaces;
    retval = copy_thread(0, clone_flags, stack_start, stack_size, p, regs);
    if (retval)
        goto bad_fork_cleanup_namespace;
@@ -1218,6 +1221,8 @@ static task_t *copy_process(unsigned lon

bad_fork_cleanup_namespace:
    exit_namespace(p);
+bad_fork_cleanup_namespaces:
+ exit_task_namespaces(p);
bad_fork_cleanup_keys:
    exit_keys(p);

```

```

bad_fork_cleanup_mm:
@@ -1559,6 +1564,7 @@ asmlinkage long sys_unshare(unsigned lon
    struct mm_struct *mm, *new_mm = NULL, *active_mm = NULL;
    struct files_struct *fd, *new_fd = NULL;
    struct sem_undo_list *new_ulist = NULL;
+ struct nsproxy *new_nsproxy, *old_nsproxy;

    check_unshare_flags(&unshare_flags);

@@ -1585,7 +1591,15 @@ asmlinkage long sys_unshare(unsigned lon

    if (new_fs || new_ns || new_sigh || new_mm || new_fd || new_ulist) {

+ old_nsproxy = current->nsproxy;
+ new_nsproxy = dup_namespaces(old_nsproxy);
+ if (!new_nsproxy) {
+   err = -ENOMEM;
+   goto bad_unshare_cleanup_semundo;
+ }
+
    task_lock(current);
+ current->nsproxy = new_nsproxy;

    if (new_fs) {
        fs = current->fs;
@@ -1621,8 +1635,10 @@ asmlinkage long sys_unshare(unsigned lon
    }

    task_unlock(current);
+ put_nsproxy(old_nsproxy);
    }

+bad_unshare_cleanup_semundo:
bad_unshare_cleanup_fd:
    if (new_fd)
        put_files_struct(new_fd);
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
new file mode 100644
index 0000000..6f53df1
--- /dev/null
+++ b/kernel/nsproxy.c
@@ -0,0 +1,78 @@
+/*
+ * Copyright (C) 2006 IBM Corporation
+ *
+ * Author: Serge Hallyn <serue@us.ibm.com>
+ *
+ * This program is free software; you can redistribute it and/or

```

```

+ * modify it under the terms of the GNU General Public License as
+ * published by the Free Software Foundation, version 2 of the
+ * License.
+ */
+
+#include <linux/compile.h>
+#include <linux/module.h>
+#include <linux/version.h>
+#include <linux/nsproxy.h>
+
+static inline void get_nsproxy(struct nsproxy *ns)
+{
+ atomic_inc(&ns->count);
+}
+
+void get_task_namespaces(struct task_struct *tsk)
+{
+ struct nsproxy *ns = tsk->nsproxy;
+ if (ns) {
+  get_nsproxy(ns);
+ }
+}
+
+/*
+ * creates a copy of "orig" with refcount 1.
+ * This does not grab references to the contained namespaces,
+ * so that needs to be done by dup_namespaces.
+ */
+static inline struct nsproxy *clone_namespaces(struct nsproxy *orig)
+{
+ struct nsproxy *ns;
+
+ ns = kmalloc(sizeof(struct nsproxy), GFP_KERNEL);
+ if (ns) {
+  memcpy(ns, orig, sizeof(struct nsproxy));
+  atomic_set(&ns->count, 1);
+ }
+ return ns;
+}
+
+/*
+ * copies the nsproxy, setting refcount to 1, and grabbing a
+ * reference to all contained namespaces. Called from
+ * sys_unshare()
+ */
+struct nsproxy *dup_namespaces(struct nsproxy *orig)
+{
+ struct nsproxy *ns = clone_namespaces(orig);

```

```

+
+ return ns;
+}
+
+/*
+ * called from clone. This now handles copy for nsproxy and all
+ * namespaces therein.
+ */
+int copy_namespaces(int flags, struct task_struct *tsk)
+{
+ struct nsproxy *old_ns = tsk->nsproxy;
+
+ if (!old_ns)
+ return 0;
+
+ get_nsproxy(old_ns);
+
+ return 0;
+}
+
+void free_nsproxy(struct nsproxy *ns)
+{
+ kfree(ns);
+}
+
+
+1.1.6

```

Subject: Re: [PATCH 1/9] namespaces: add nsproxy
 Posted by [Sam Vilain](#) on Sun, 21 May 2006 23:30:05 GMT
[View Forum Message](#) <> [Reply to Message](#)

Serge E. Hallyn wrote:

```

>@@ -1585,7 +1591,15 @@ asmlinkage long sys_unshare(unsigned lon
>
> if (new_fs || new_ns || new_sigh || new_mm || new_fd || new_ulist) {
>
>+ old_nsproxy = current->nsproxy;
>+ new_nsproxy = dup_namespaces(old_nsproxy);
>+ if (!new_nsproxy) {
>+ err = -ENOMEM;
>+ goto bad_unshare_cleanup_semundo;
>+ }
>+
> task_lock(current);
>
>

```


We'll get lots of duplicate nsproxy structures before we move all of the pointers for those subsystems into it. Do we need to dup namespaces on all of those conditions?

Sam.

Subject: Re: [PATCH 1/9] namespaces: add nsproxy
Posted by [ebiederm](#) on Sun, 21 May 2006 23:38:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

Sam Vilain <sam@vilain.net> writes:

```
> Serge E. Hallyn wrote:
>
>>@@ -1585,7 +1591,15 @@ asmlinkage long sys_unshare(unsigned lon
>>
>> if (new_fs || new_ns || new_sigh || new_mm || new_fd || new_ulist) {
>>
>>+ old_nsproxy = current->nsproxy;
>>+ new_nsproxy = dup_namespaces(old_nsproxy);
>>+ if (!new_nsproxy) {
>>+   err = -ENOMEM;
>>+   goto bad_unshare_cleanup_semundo;
>>+ }
>>+
>> task_lock(current);
>>
>>
>
> We'll get lots of duplicate nsproxy structures before we move all of the
> pointers for those subsystems into it. Do we need to dup namespaces on
> all of those conditions?
```

Ugh. Good catch. The new nsproxy needs to be just for the fs and the uts namespace.

I guess that means that test should be moved up a few lines.

Eric

Subject: Re: [PATCH 1/9] namespaces: add nsproxy
Posted by [serue](#) on Mon, 22 May 2006 12:39:19 GMT
[View Forum Message](#) <> [Reply to Message](#)

Quoting Eric W. Biederman (ebiederm@xmission.com):

> Sam Vilain <sam@vilain.net> writes:

>

> > Serge E. Hallyn wrote:

> >

> >>@@ -1585,7 +1591,15 @@ asmlinkage long sys_unshare(unsigned lon

> >>

> >> if (new_fs || new_ns || new_sigh || new_mm || new_fd || new_ulist) {

> >>

> >>+ old_nsproxy = current->nsproxy;

> >>+ new_nsproxy = dup_namespaces(old_nsproxy);

> >>+ if (!new_nsproxy) {

> >>+ err = -ENOMEM;

> >>+ goto bad_unshare_cleanup_semundo;

> >>+ }

> >>+

> >> task_lock(current);

> >>

> >>

> >

> > We'll get lots of duplicate nsproxy structures before we move all of the

> > pointers for those subsystems into it. Do we need to dup namespaces on

> > all of those conditions?

>

> Ugh. Good catch. The new nsproxy needs to be just for the fs and the uts
> namespace.

>

> I guess that means that test should be moved up a few lines.

Oh. Yeah. It didn't look odd to me bc it's about the number of
namespaces we are *going* to have :)

Fix follows:

Subject: [PATCH] uts: copy nsproxy only when needed.

From: Serge Hallyn <serue@us.ibm.com>

The nsproxy was being copied in unshare() when anything was being
unshared, even if it was something not referenced from nsproxy.
This should end up in some cases with far more memory usage than
necessary.

Signed-off-by: Serge Hallyn <serue@us.ibm.com>

kernel/fork.c | 20 ++++++++-----

1 files changed, 14 insertions(+), 6 deletions(-)

```

74d1068458c62302ac8ed38e38a57b692580662f
diff --git a/kernel/fork.c b/kernel/fork.c
index cdc549e..9278a68 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1559,7 +1559,7 @@ asmlinkage long sys_unshare(unsigned lon
    struct mm_struct *mm, *new_mm = NULL, *active_mm = NULL;
    struct files_struct *fd, *new_fd = NULL;
    struct sem_undo_list *new_ulist = NULL;
- struct nsproxy *new_nsproxy, *old_nsproxy;
+ struct nsproxy *new_nsproxy = NULL, *old_nsproxy = NULL;
    struct uts_namespace *uts, *new_uts = NULL;

    check_unshare_flags(&unshare_flags);
@@ -1587,18 +1587,24 @@ asmlinkage long sys_unshare(unsigned lon
    if ((err = unshare_utsname(unshare_flags, &new_uts)))
        goto bad_unshare_cleanup_semundo;

- if (new_fs || new_ns || new_sigh || new_mm || new_fd || new_ulist ||
-     new_uts) {
-
+ if (new_ns || new_uts) {
+     old_nsproxy = current->nsproxy;
+     new_nsproxy = dup_namespaces(old_nsproxy);
+     if (!new_nsproxy) {
+         err = -ENOMEM;
+         goto bad_unshare_cleanup_uts;
+     }
+ }
+
+ if (new_fs || new_ns || new_sigh || new_mm || new_fd || new_ulist ||
+     new_uts) {

    task_lock(current);
- current->nsproxy = new_nsproxy;
+
+ if (new_nsproxy) {
+     current->nsproxy = new_nsproxy;
+     new_nsproxy = old_nsproxy;
+ }

    if (new_fs) {
        fs = current->fs;
@@ -1640,9 +1640,11 @@ asmlinkage long sys_unshare(unsigned lon
    }

    task_unlock(current);

```

```
- put_nsproxy(old_nsproxy);
}

+ if (new_nsproxy)
+ put_nsproxy(new_nsproxy);
+
bad_unshare_cleanup_uts:
if (new_uts)
put_uts_ns(new_uts);
--
```

1.1.6
