Subject: namespace and nsproxy syscalls
Posted by Cedric Le Goater on Tue, 26 Sep 2006 09:42:10 GMT
View Forum Message <> Reply to Message

Hello all,

A while ago, we expressed the need to have a new syscall specific to
namespaces. the clone and unshare are good candidates but we are reaching
the limit of the clone flags and clone has been hijacked enough.

So, I came up with unshare_ns. the patch for the core feature follows
the email. Not much difference with unshare() for the moment but it gives
us the freedom to diverge when new namespaces come in. I have faith also !
If you feel it's useful, i'll send the full patchset for review on the list.

I'd like to discuss of another syscall which would allow a process to
bind to a set of namespaces (== nsproxy == container) :

 bind_ns(ns_id_t id, int flags)

bind_ns binds the current nsproxy to an id. You can only bind once and
you can use this id to bind another process to the same nsproxy.

a few comments :

* ns_id_t could be an int, a const char*, a struct ns_addr*. this is
  to be defined.
* bind_ns applies to nsproxy and not to namespaces. we could bind a
  specific namespace to an id using flags but i don't see the need.
  (but why not with a flags 0 defining ALL namespaces)
* semantic is close to shmat but i don't think we need a shmdt because
  nsproxies are not resilient objects.

Thanks,

C.

From: Cedric Le Goater <clg@fr.ibm.com>
Subject: add unshare_ns syscall core routine

This patch adds the unshare_ns syscall core routine.

This syscall is an unshare dedicated to namespaces.

sample user program :

/*

```
 * unshare_ns.c
 *
 * author: Cedric Le Goater <clg@fr.ibm.com>
 *
 * (C) Copyright IBM Corp. 2005, 2006
 *
 * 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; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
 * 02110-1301, USA.
 */

#include <stdio.h>
#include <stdlib.h>
#include <sched.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#include <linux/unistd.h>

#ifndef __NR_unshare
#if __i386__
#    define __NR_unshare_ns 319
#elif __x86_64__
#    define __NR_unshare_ns 280
#elif __ia64__
#    define __NR_unshare_ns 1303
#elif __s390x__
#    define __NR_unshare_ns 310
#elif __powerpc__
#    define __NR_unshare_ns 301
#else
#    error "Architecture not supported"
#endif
#endif

static inline _syscall1 (int,  unshare_ns, int, flags)
```

```
#define UNSHARE_NS_MNT  0x00000001
#define UNSHARE_NS_UTS  0x00000002
#define UNSHARE_NS_IPC  0x00000004
#define UNSHARE_NS_USER  0x00000008
#define UNSHARE_NS_NET  0x00000010
#define UNSHARE_NS_PID  0x00000020

static void usage(const char *name)
{
 printf("usage: %s [-Hiunmpeh]\n", name);
 printf("\t-H : unshare utsname namespace.\n");
 printf("\t-i : unshare ipc namespace.\n");
 printf("\t-u : unshare user namespace.\n");
 printf("\t-n : unshare net namespace.\n");
 printf("\t-m : unshare mount namespace.\n");
 printf("\t-p : unshare pid namespace.\n");
 printf("\t-e : exec command.\n");
 printf("\n");
 printf("(C) Copyright IBM Corp. 2005, 2006\n");
 printf("\n");
 exit(1);
}

int main(int argc, char* argv[])
{
 int c;
 unsigned long flag = 0;
 int exec = 0;

 while ((c = getopt(argc, argv, "+Hiunmpeh")) != EOF) {
  switch (c) {
  case 'i': flag |= UNSHARE_NS_IPC; break;
  case 'u': flag |= UNSHARE_NS_USER; break;
  case 'H': flag |= UNSHARE_NS_UTS; break;
  case 'n': flag |= UNSHARE_NS_NET; break;
  case 'm': flag |= UNSHARE_NS_MNT; break;
  case 'p': flag |= UNSHARE_NS_PID; break;
  case 'e': exec = 1; break;
  case 'h':
  default:
   usage(argv[0]);
  }
 };

 argv = &argv[optind];
 argc = argc - optind;
```

```
 if (unshare_ns(flag) == -1) {
  perror("unshare_ns");
  return 1;
 }

 if (exec) {
  execve(argv[0], argv, __environ);
  fprintf(stderr, "execve(%s) : %s\n", argv[0], strerror(errno));
  return 1;
 }

 return 0;
}
```

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

```
---
 include/linux/sched.h    |   10 ++++
 include/linux/syscalls.h |    1
 kernel/fork.c            |  106 +++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 117 insertions(+)

Index: 2.6.18-mm1/include/linux/sched.h
===================================================================
--- 2.6.18-mm1.orig/include/linux/sched.h
+++ 2.6.18-mm1/include/linux/sched.h
@@ -28,6 +28,16 @@
 #define CLONE_NEWIPC  0x08000000 /* New ipcs */

 /*
+ * unshare_ns flags:
+ */
+#define UNSHARE_NS_MNT  0x00000001
+#define UNSHARE_NS_UTS  0x00000002
+#define UNSHARE_NS_IPC  0x00000004
+#define UNSHARE_NS_USER  0x00000008
+#define UNSHARE_NS_NET  0x00000010
+#define UNSHARE_NS_PID  0x00000020
+
+/*
  * Scheduling policies
  */
 #define SCHED_NORMAL  0
Index: 2.6.18-mm1/include/linux/syscalls.h
===================================================================
--- 2.6.18-mm1.orig/include/linux/syscalls.h
+++ 2.6.18-mm1/include/linux/syscalls.h
```

```
@@ -580,6 +580,7 @@ asmlinkage long compat_sys_newfstatat(un
 asmlinkage long compat_sys_openat(unsigned int dfd, const char __user *filename,
        int flags, int mode);
 asmlinkage long sys_unshare(unsigned long unshare_flags);
+asmlinkage long sys_unshare_ns(unsigned long unshare_flags);

 asmlinkage long sys_splice(int fd_in, loff_t __user *off_in,
       int fd_out, loff_t __user *off_out,
Index: 2.6.18-mm1/kernel/fork.c
===================================================================
--- 2.6.18-mm1.orig/kernel/fork.c
+++ 2.6.18-mm1/kernel/fork.c
@@ -1761,3 +1761,109 @@ bad_unshare_cleanup_thread:
 bad_unshare_out:
  return err;
 }
+
+/*
+ * unshare_ns allows a process to 'unshare' one or more of its
+ * namespaces which were originally shared using clone.
+ */
+asmlinkage long sys_unshare_ns(unsigned long unshare_ns_flags)
+{
+ int err = 0;
+ struct nsproxy *new_nsproxy = NULL, *old_nsproxy = NULL;
+ struct fs_struct *fs, *new_fs = NULL;
+ struct mnt_namespace *mnt, *new_mnt = NULL;
+ struct uts_namespace *uts, *new_uts = NULL;
+ struct ipc_namespace *ipc, *new_ipc = NULL;
+ unsigned long unshare_flags = 0;
+
+ /* Return -EINVAL for all unsupported flags */
+ err = -EINVAL;
+ if (unshare_ns_flags & ~(UNSHARE_NS_MNT|UNSHARE_NS_UTS|UNSHARE_NS_IPC|
+     UNSHARE_NS_USER|UNSHARE_NS_NET|
+     UNSHARE_NS_PID))
+  goto bad_unshare_ns_out;
+
+ /* convert unshare_ns flags to clone flags */
+ if (unshare_ns_flags & UNSHARE_NS_MNT)
+  unshare_flags |= CLONE_NEWNS|CLONE_FS;
+ if (unshare_ns_flags & UNSHARE_NS_UTS)
+  unshare_flags |= CLONE_NEWUTS;
+ if (unshare_ns_flags & UNSHARE_NS_IPC)
+  unshare_flags |= CLONE_NEWIPC;
+
+ if ((err = unshare_fs(unshare_flags, &new_fs)))
+  goto bad_unshare_ns_out;
```

```
+ if ((err = unshare_mnt_namespace(unshare_flags, &new_mnt, new_fs)))
+  goto bad_unshare_ns_cleanup_fs;
+ if ((err = unshare_utsname(unshare_flags, &new_uts)))
+  goto bad_unshare_ns_cleanup_mnt;
+ if ((err = unshare_ipcs(unshare_flags, &new_ipc)))
+  goto bad_unshare_ns_cleanup_uts;
+
+ if (new_mnt || new_uts || new_ipc) {
+  old_nsproxy = current->nsproxy;
+  new_nsproxy = dup_namespaces(old_nsproxy);
+  if (!new_nsproxy) {
+   err = -ENOMEM;
+   goto bad_unshare_ns_cleanup_ipc;
+  }
+ }
+
+ if (new_fs || new_mnt || new_uts || new_ipc) {
+
+  task_lock(current);
+
+  if (new_nsproxy) {
+   current->nsproxy = new_nsproxy;
+   new_nsproxy = old_nsproxy;
+  }
+
+  if (new_fs) {
+   fs = current->fs;
+   current->fs = new_fs;
+   new_fs = fs;
+  }
+
+  if (new_mnt) {
+   mnt = current->nsproxy->mnt_ns;
+   current->nsproxy->mnt_ns = new_mnt;
+   new_mnt = mnt;
+  }
+
+  if (new_uts) {
+   uts = current->nsproxy->uts_ns;
+   current->nsproxy->uts_ns = new_uts;
+   new_uts = uts;
+  }
+
+  if (new_ipc) {
+   ipc = current->nsproxy->ipc_ns;
+   current->nsproxy->ipc_ns = new_ipc;
+   new_ipc = ipc;
+  }
```

```
+
+  task_unlock(current);
+ }
+
+ if (new_nsproxy)
+  put_nsproxy(new_nsproxy);
+
+bad_unshare_ns_cleanup_ipc:
+ if (new_ipc)
+  put_ipc_ns(new_ipc);
+
+bad_unshare_ns_cleanup_uts:
+ if (new_uts)
+  put_uts_ns(new_uts);
+
+bad_unshare_ns_cleanup_mnt:
+ if (new_mnt)
+  put_mnt_ns(new_mnt);
+
+bad_unshare_ns_cleanup_fs:
+ if (new_fs)
+  put_fs_struct(new_fs);
+
+bad_unshare_ns_out:
+ return err;
+}
```

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers