

---

Subject: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [Sukadev Bhattiprolu](#) on Wed, 09 Apr 2008 22:26:11 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

This is a resend of the patch set Cedric had sent earlier. I ported the patch set to 2.6.25-rc8-mm1 and tested on x86 and x86\_64.

---

We have run out of the 32 bits in clone\_flags !

This patchset introduces 2 new system calls which support 64bit clone-flags.

```
long sys_clone64(unsigned long flags_high, unsigned long flags_low,  
unsigned long newsp);
```

```
long sys_unshare64(unsigned long flags_high, unsigned long flags_low);
```

The current version of clone64() does not support CLONE\_PARENT\_SETTID and CLONE\_CHILD\_CLEARTID because we would exceed the 6 registers limit of some arches. It's possible to get around this limitation but we might not need it as we already have clone()

This is work in progress but already includes support for x86, x86\_64, x86\_64(32), ppc64, ppc64(32), s390x, s390x(31).

ia64 already supports 64bits clone flags through the clone2() syscall. should we harmonize the name to clone2 ?

---

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

---

---

Subject: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Sukadev Bhattiprolu](#) on Wed, 09 Apr 2008 22:32:31 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>  
Subject: [lxc-dev] [patch -lxc 1/3] change clone\_flags type to u64

This is a preliminary patch changing the clone\_flags type to 64bits for all the routines called by do\_fork().

It prepares ground for the next patch which introduces an enhanced version of clone() supporting 64bits flags.

This is work in progress. All conversions might not be done yet.

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

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

---

```
arch/alpha/kernel/process.c      | 2 +-
arch/arm/kernel/process.c        | 2 +-
arch/avr32/kernel/process.c      | 2 +-
arch/blackfin/kernel/process.c   | 2 +-
arch/cris/arch-v10/kernel/process.c | 2 +-
arch/cris/arch-v32/kernel/process.c | 2 +-
arch/frv/kernel/process.c        | 2 +-
arch/h8300/kernel/process.c      | 2 +-
arch/ia64/ia32/sys_ia32.c        | 2 +-
arch/ia64/kernel/process.c       | 2 +-
arch/m32r/kernel/process.c       | 2 +-
arch/m68k/kernel/process.c       | 2 +-
arch/m68knommu/kernel/process.c  | 2 +-
arch/mips/kernel/process.c       | 2 +-
arch/mn10300/kernel/process.c    | 2 +-
arch/parisc/kernel/process.c     | 2 +-
arch/powerpc/kernel/process.c    | 2 +-
arch/s390/kernel/process.c       | 2 +-
arch/sh/kernel/process_32.c      | 2 +-
arch/sh/kernel/process_64.c     | 2 +-
arch/sparc/kernel/process.c      | 2 +-
arch/sparc64/kernel/process.c    | 2 +-
arch/um/kernel/process.c         | 2 +-
arch/v850/kernel/process.c       | 2 +-
arch/x86/kernel/process_32.c     | 2 +-
arch/x86/kernel/process_64.c    | 2 +-
arch/xtensa/kernel/process.c     | 2 +-
fs/namespace.c                  | 2 +-
include/linux/ipc_namespace.h    | 4 +++-
include/linux/key.h              | 2 +-
include/linux/mnt_namespace.h    | 2 +-
include/linux/nsproxy.h         | 4 +++-
include/linux/pid_namespace.h    | 4 +++-
include/linux/sched.h           | 6 +++++-
include/linux/security.h        | 6 +++++-
include/linux/sem.h              | 4 +++-
include/linux/user_namespace.h   | 4 +++-
include/linux/utsname.h         | 4 +++-
include/net/net_namespace.h      | 4 +++-
ipc/namespace.c                 | 2 +-
ipc/sem.c                       | 2 +-
kernel/fork.c                    | 36 ++++++-----
kernel/nsproxy.c                 | 6 +++++-
kernel/pid_namespace.c          | 2 +-
kernel/user_namespace.c         | 2 +-

```

```

kernel/utsname.c          | 2 +-
net/core/net_namespace.c | 4 +++-
security/dummy.c         | 2 +-
security/keys/process_keys.c | 2 +-
security/security.c      | 2 +-
security/selinux/hooks.c | 2 +-
51 files changed, 83 insertions(+), 81 deletions(-)

```

Index: 2.6.25-rc2-mm1/arch/alpha/kernel/process.c

```

=====
--- 2.6.25-rc2-mm1.orig/arch/alpha/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/alpha/kernel/process.c
@@ -270,7 +270,7 @@ alpha_vfork(struct pt_regs *regs)
 */

```

int

```

-copy_thread(int nr, unsigned long clone_flags, unsigned long usp,
+copy_thread(int nr, u64 clone_flags, unsigned long usp,
             unsigned long unused,
             struct task_struct * p, struct pt_regs * regs)

```

{

Index: 2.6.25-rc2-mm1/arch/arm/kernel/process.c

```

=====
--- 2.6.25-rc2-mm1.orig/arch/arm/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/arm/kernel/process.c
@@ -331,7 +331,7 @@ void release_thread(struct task_struct *
asmlinkage void ret_from_fork(void) __asm__("ret_from_fork");

```

int

```

-copy_thread(int nr, unsigned long clone_flags, unsigned long stack_start,
+copy_thread(int nr, u64 clone_flags, unsigned long stack_start,
             unsigned long stk_sz, struct task_struct *p, struct pt_regs *regs)

```

{

```

    struct thread_info *thread = task_thread_info(p);

```

Index: 2.6.25-rc2-mm1/arch/avr32/kernel/process.c

```

=====
--- 2.6.25-rc2-mm1.orig/arch/avr32/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/avr32/kernel/process.c
@@ -325,7 +325,7 @@ int dump_fpu(struct pt_regs *regs, elf_f

```

```

asmlinkage void ret_from_fork(void);

```

```

-int copy_thread(int nr, unsigned long clone_flags, unsigned long usp,
+int copy_thread(int nr, u64 clone_flags, unsigned long usp,
                unsigned long unused,
                struct task_struct *p, struct pt_regs *regs)

```

{

Index: 2.6.25-rc2-mm1/arch/blackfin/kernel/process.c

```

=====
--- 2.6.25-rc2-mm1.orig/arch/blackfin/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/blackfin/kernel/process.c
@@ -168,7 +168,7 @@ asmlinkage int bfin_clone(struct pt_regs
}

int
-copy_thread(int nr, unsigned long clone_flags,
+copy_thread(int nr, u64 clone_flags,
    unsigned long usp, unsigned long topstk,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/cris/arch-v10/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/cris/arch-v10/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/cris/arch-v10/kernel/process.c
@@ -115,7 +115,7 @@ int kernel_thread(int (*fn)(void *), voi
*/
asmlinkage void ret_from_fork(void);

-int copy_thread(int nr, unsigned long clone_flags, unsigned long usp,
+int copy_thread(int nr, u64 clone_flags, unsigned long usp,
    unsigned long unused,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/cris/arch-v32/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/cris/arch-v32/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/cris/arch-v32/kernel/process.c
@@ -131,7 +131,7 @@ kernel_thread(int (*fn)(void *), void *
extern asmlinkage void ret_from_fork(void);

int
-copy_thread(int nr, unsigned long clone_flags, unsigned long usp,
+copy_thread(int nr, u64 clone_flags, unsigned long usp,
    unsigned long unused,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/frv/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/frv/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/frv/kernel/process.c
@@ -204,7 +204,7 @@ void prepare_to_copy(struct task_struct
/*
 * set up the kernel stack and exception frames for a new process
 */
-int copy_thread(int nr, unsigned long clone_flags,
+int copy_thread(int nr, u64 clone_flags,

```

```

    unsigned long usp, unsigned long topstk,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/h8300/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/h8300/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/h8300/kernel/process.c
@@ -192,7 +192,7 @@ asmlinkage int h8300_clone(struct pt_reg

}

```

```

-int copy_thread(int nr, unsigned long clone_flags,
+int copy_thread(int nr, u64 clone_flags,
    unsigned long usp, unsigned long topstk,
    struct task_struct * p, struct pt_regs * regs)
{
Index: 2.6.25-rc2-mm1/arch/ia64/ia32/sys_ia32.c
=====
--- 2.6.25-rc2-mm1.orig/arch/ia64/ia32/sys_ia32.c
+++ 2.6.25-rc2-mm1/arch/ia64/ia32/sys_ia32.c
@@ -734,7 +734,7 @@ __ia32_copy_pp_list(struct ia64_partial_

```

```

int
ia32_copy_ia64_partial_page_list(struct task_struct *p,
- unsigned long clone_flags)
+ u64 clone_flags)
{
    int retval = 0;

```

```

Index: 2.6.25-rc2-mm1/arch/ia64/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/ia64/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/ia64/kernel/process.c
@@ -402,7 +402,7 @@ ia64_load_extra (struct task_struct *tas
 * so there is nothing to worry about.
 */
int
-copy_thread (int nr, unsigned long clone_flags,
+copy_thread(int nr, u64 clone_flags,
    unsigned long user_stack_base, unsigned long user_stack_size,
    struct task_struct *p, struct pt_regs *regs)
{

```

```

Index: 2.6.25-rc2-mm1/arch/m32r/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/m32r/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/m32r/kernel/process.c
@@ -242,7 +242,7 @@ int dump_fpu(struct pt_regs *regs, elf_f
    return 0; /* Task didn't use the fpu at all. */

```

```
}
```

```
-int copy_thread(int nr, unsigned long clone_flags, unsigned long spu,  
+int copy_thread(int nr, u64 clone_flags, unsigned long spu,  
    unsigned long unused, struct task_struct *tsk, struct pt_regs *regs)
```

```
{  
    struct pt_regs *childregs = task_pt_regs(tsk);
```

```
Index: 2.6.25-rc2-mm1/arch/m68k/kernel/process.c
```

```
-----  
--- 2.6.25-rc2-mm1.orig/arch/m68k/kernel/process.c
```

```
+++ 2.6.25-rc2-mm1/arch/m68k/kernel/process.c
```

```
@@ -235,7 +235,7 @@ asmlinkage int m68k_clone(struct pt_regs  
    parent_tidptr, child_tidptr);
```

```
}
```

```
-int copy_thread(int nr, unsigned long clone_flags, unsigned long usp,  
+int copy_thread(int nr, u64 clone_flags, unsigned long usp,  
    unsigned long unused,  
    struct task_struct * p, struct pt_regs * regs)
```

```
{  
Index: 2.6.25-rc2-mm1/arch/m68knommu/kernel/process.c
```

```
-----  
--- 2.6.25-rc2-mm1.orig/arch/m68knommu/kernel/process.c
```

```
+++ 2.6.25-rc2-mm1/arch/m68knommu/kernel/process.c
```

```
@@ -200,7 +200,7 @@ asmlinkage int m68k_clone(struct pt_regs  
    return do_fork(clone_flags, newsp, regs, 0, NULL, NULL);
```

```
}
```

```
-int copy_thread(int nr, unsigned long clone_flags,  
+int copy_thread(int nr, u64 clone_flags,  
    unsigned long usp, unsigned long topstk,  
    struct task_struct * p, struct pt_regs * regs)
```

```
{  
Index: 2.6.25-rc2-mm1/arch/mips/kernel/process.c
```

```
-----  
--- 2.6.25-rc2-mm1.orig/arch/mips/kernel/process.c
```

```
+++ 2.6.25-rc2-mm1/arch/mips/kernel/process.c
```

```
@@ -100,7 +100,7 @@ void flush_thread(void)
```

```
{
```

```
}
```

```
-int copy_thread(int nr, unsigned long clone_flags, unsigned long usp,  
+int copy_thread(int nr, u64 clone_flags, unsigned long usp,  
    unsigned long unused, struct task_struct *p, struct pt_regs *regs)
```

```
{  
    struct thread_info *ti = task_thread_info(p);
```

```
Index: 2.6.25-rc2-mm1/arch/mn10300/kernel/process.c
```

```

--- 2.6.25-rc2-mm1.orig/arch/mn10300/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/mn10300/kernel/process.c
@@ -193,7 +193,7 @@ void prepare_to_copy(struct task_struct
 * set up the kernel stack for a new thread and copy arch-specific thread
 * control information
 */

```

```

-int copy_thread(int nr, unsigned long clone_flags,
+int copy_thread(int nr, u64 clone_flags,
    unsigned long c_esp, unsigned long ustk_size,
    struct task_struct *p, struct pt_regs *kregs)

```

```

{
Index: 2.6.25-rc2-mm1/arch/parisc/kernel/process.c

```

```

=====
--- 2.6.25-rc2-mm1.orig/arch/parisc/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/parisc/kernel/process.c
@@ -263,7 +263,7 @@ sys_vfork(struct pt_regs *regs)
}

```

```

int
-copy_thread(int nr, unsigned long clone_flags, unsigned long esp,
+copy_thread(int nr, u64 clone_flags, unsigned long esp,
    unsigned long unused, /* in ia64 this is "user_stack_size" */
    struct task_struct * p, struct pt_regs * pregs)

```

```

{
Index: 2.6.25-rc2-mm1/arch/powerpc/kernel/process.c

```

```

=====
--- 2.6.25-rc2-mm1.orig/arch/powerpc/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/powerpc/kernel/process.c
@@ -534,7 +534,7 @@ void prepare_to_copy(struct task_struct
/*
 * Copy a thread..
 */

```

```

-int copy_thread(int nr, unsigned long clone_flags, unsigned long esp,
+int copy_thread(int nr, u64 clone_flags, unsigned long esp,
    unsigned long unused, struct task_struct *p,
    struct pt_regs *regs)

```

```

{
Index: 2.6.25-rc2-mm1/arch/s390/kernel/process.c

```

```

=====
--- 2.6.25-rc2-mm1.orig/arch/s390/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/s390/kernel/process.c
@@ -243,7 +243,7 @@ void release_thread(struct task_struct *
{
}

```

```

-int copy_thread(int nr, unsigned long clone_flags, unsigned long new_stackp,
+int copy_thread(int nr, u64 clone_flags, unsigned long new_stackp,
    unsigned long unused,

```

```

    struct task_struct * p, struct pt_regs * regs)
{
Index: 2.6.25-rc2-mm1/arch/sh/kernel/process_32.c
=====
--- 2.6.25-rc2-mm1.orig/arch/sh/kernel/process_32.c
+++ 2.6.25-rc2-mm1/arch/sh/kernel/process_32.c
@@ -232,7 +232,7 @@ int dump_fpu(struct pt_regs *regs, elf_f

asmlinkage void ret_from_fork(void);

-int copy_thread(int nr, unsigned long clone_flags, unsigned long usp,
+int copy_thread(int nr, u64 clone_flags, unsigned long usp,
    unsigned long unused,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/sh/kernel/process_64.c
=====
--- 2.6.25-rc2-mm1.orig/arch/sh/kernel/process_64.c
+++ 2.6.25-rc2-mm1/arch/sh/kernel/process_64.c
@@ -500,7 +500,7 @@ int dump_fpu(struct pt_regs *regs, elf_f

asmlinkage void ret_from_fork(void);

-int copy_thread(int nr, unsigned long clone_flags, unsigned long usp,
+int copy_thread(int nr, u64 clone_flags, unsigned long usp,
    unsigned long unused,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/sparc/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/sparc/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/sparc/kernel/process.c
@@ -454,7 +454,7 @@ asmlinkage int sparc_do_fork(unsigned lo
*/
extern void ret_from_fork(void);

-int copy_thread(int nr, unsigned long clone_flags, unsigned long sp,
+int copy_thread(int nr, u64 clone_flags, unsigned long sp,
    unsigned long unused,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/sparc64/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/sparc64/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/sparc64/kernel/process.c
@@ -617,7 +617,7 @@ asmlinkage long sparc_do_fork(unsigned l
* Parent --> %o0 == childs pid, %o1 == 0
* Child --> %o0 == parents pid, %o1 == 1

```



```

*/
-int copy_thread(int nr, unsigned long clone_flags, unsigned long sp,
+int copy_thread(int nr, u64 clone_flags, unsigned long sp,
    unsigned long unused,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/um/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/um/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/um/kernel/process.c
@@ -181,7 +181,7 @@ void fork_handler(void)
    userspace(&current->thread.regs.regs);
}

-int copy_thread(int nr, unsigned long clone_flags, unsigned long sp,
+int copy_thread(int nr, u64 clone_flags, unsigned long sp,
    unsigned long stack_top, struct task_struct * p,
    struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/v850/kernel/process.c
=====
--- 2.6.25-rc2-mm1.orig/arch/v850/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/v850/kernel/process.c
@@ -110,7 +110,7 @@ void flush_thread (void)
    set_fs (USER_DS);
}

-int copy_thread (int nr, unsigned long clone_flags,
+int copy_thread(int nr, u64 clone_flags,
    unsigned long stack_start, unsigned long stack_size,
    struct task_struct *p, struct pt_regs *regs)
{
Index: 2.6.25-rc2-mm1/arch/x86/kernel/process_32.c
=====
--- 2.6.25-rc2-mm1.orig/arch/x86/kernel/process_32.c
+++ 2.6.25-rc2-mm1/arch/x86/kernel/process_32.c
@@ -494,7 +494,7 @@ void prepare_to_copy(struct task_struct
    unlazy_fpu(tsk);
}

-int copy_thread(int nr, unsigned long clone_flags, unsigned long sp,
+int copy_thread(int nr, u64 clone_flags, unsigned long sp,
    unsigned long unused,
    struct task_struct * p, struct pt_regs * regs)
{
Index: 2.6.25-rc2-mm1/arch/x86/kernel/process_64.c
=====
--- 2.6.25-rc2-mm1.orig/arch/x86/kernel/process_64.c

```

```
+++ 2.6.25-rc2-mm1/arch/x86/kernel/process_64.c
@@ -491,7 +491,7 @@ void prepare_to_copy(struct task_struct
    unlazy_fpu(tsk);
}
```

```
-int copy_thread(int nr, unsigned long clone_flags, unsigned long sp,
+int copy_thread(int nr, u64 clone_flags, unsigned long sp,
    unsigned long unused,
    struct task_struct * p, struct pt_regs * regs)
{
```

Index: 2.6.25-rc2-mm1/arch/xtensa/kernel/process.c

```
--- 2.6.25-rc2-mm1.orig/arch/xtensa/kernel/process.c
+++ 2.6.25-rc2-mm1/arch/xtensa/kernel/process.c
@@ -172,7 +172,7 @@ void prepare_to_copy(struct task_struct
 *    childregs.
 */
```

```
-int copy_thread(int nr, unsigned long clone_flags, unsigned long usp,
+int copy_thread(int nr, u64 clone_flags, unsigned long usp,
    unsigned long unused,
    struct task_struct * p, struct pt_regs * regs)
{
```

Index: 2.6.25-rc2-mm1/include/linux/key.h

```
--- 2.6.25-rc2-mm1.orig/include/linux/key.h
+++ 2.6.25-rc2-mm1/include/linux/key.h
@@ -269,7 +269,7 @@ extern struct key root_user_keyring, roo
extern int alloc_uid_keyring(struct user_struct *user,
    struct task_struct *ctx);
extern void switch_uid_keyring(struct user_struct *new_user);
-extern int copy_keys(unsigned long clone_flags, struct task_struct *tsk);
+extern int copy_keys(u64 clone_flags, struct task_struct *tsk);
extern int copy_thread_group_keys(struct task_struct *tsk);
extern void exit_keys(struct task_struct *tsk);
extern void exit_thread_group_keys(struct signal_struct *tg);
```

Index: 2.6.25-rc2-mm1/include/linux/sched.h

```
--- 2.6.25-rc2-mm1.orig/include/linux/sched.h
+++ 2.6.25-rc2-mm1/include/linux/sched.h
@@ -1761,7 +1761,8 @@ extern struct mm_struct *get_task_mm(str
/* Remove the current tasks stale references to the old mm_struct */
extern void mm_release(struct task_struct *, struct mm_struct *);
```

```
-extern int copy_thread(int, unsigned long, unsigned long, unsigned long, struct task_struct *,
struct pt_regs *);
+extern int copy_thread(int, u64, unsigned long, unsigned long,
+    struct task_struct *, struct pt_regs *);
```

```
extern void flush_thread(void);
extern void exit_thread(void);
```

```
@@ -1777,7 +1778,8 @@ extern int allow_signal(int);
extern int disallow_signal(int);
```

```
extern int do_execve(char *, char __user * __user *, char __user * __user *, struct pt_regs *);
-extern long do_fork(unsigned long, unsigned long, struct pt_regs *, unsigned long, int __user *,
int __user *);
+extern long do_fork(u64, unsigned long, struct pt_regs *, unsigned long,
+ int __user *, int __user *);
struct task_struct *fork_idle(int);
```

```
extern void set_task_comm(struct task_struct *tsk, char *from);
Index: 2.6.25-rc2-mm1/include/linux/security.h
```

```
=====
--- 2.6.25-rc2-mm1.orig/include/linux/security.h
+++ 2.6.25-rc2-mm1/include/linux/security.h
@@ -1332,7 +1332,7 @@ struct security_operations {
    int (*file_receive) (struct file * file);
    int (*dentry_open) (struct file *file);

- int (*task_create) (unsigned long clone_flags);
+ int (*task_create) (u64 clone_flags);
    int (*task_alloc_security) (struct task_struct * p);
    void (*task_free_security) (struct task_struct * p);
    int (*task_setuid) (uid_t id0, uid_t id1, uid_t id2, int flags);
@@ -1587,7 +1587,7 @@ int security_file_send_sigiotask(struct
    struct fown_struct *fown, int sig);
int security_file_receive(struct file *file);
int security_dentry_open(struct file *file);
-int security_task_create(unsigned long clone_flags);
+int security_task_create(u64 clone_flags);
int security_task_alloc(struct task_struct *p);
void security_task_free(struct task_struct *p);
int security_task_setuid(uid_t id0, uid_t id1, uid_t id2, int flags);
@@ -2053,7 +2053,7 @@ static inline int security_dentry_open (
    return 0;
}

-static inline int security_task_create (unsigned long clone_flags)
+static inline int security_task_create(u64 clone_flags)
{
    return 0;
}
```

```
Index: 2.6.25-rc2-mm1/include/linux/sem.h
```

```
=====
--- 2.6.25-rc2-mm1.orig/include/linux/sem.h
```

```

+++ 2.6.25-rc2-mm1/include/linux/sem.h
@@ -139,11 +139,11 @@ struct sysv_sem {

#ifdef CONFIG_SYSVIPC

-extern int copy_semundo(unsigned long clone_flags, struct task_struct *tsk);
+extern int copy_semundo(u64 clone_flags, struct task_struct *tsk);
extern void exit_sem(struct task_struct *tsk);

#else
-static inline int copy_semundo(unsigned long clone_flags, struct task_struct *tsk)
+static inline int copy_semundo(u64 clone_flags, struct task_struct *tsk)
{
    return 0;
}
Index: 2.6.25-rc2-mm1/ipc/sem.c
=====
--- 2.6.25-rc2-mm1.orig/ipc/sem.c
+++ 2.6.25-rc2-mm1/ipc/sem.c
@@ -1212,7 +1212,7 @@ asmlinkage long sys_semop (int semid, st
 * parent and child tasks.
 */

-int copy_semundo(unsigned long clone_flags, struct task_struct *tsk)
+int copy_semundo(u64 clone_flags, struct task_struct *tsk)
{
    struct sem_undo_list *undo_list;
    int error;
Index: 2.6.25-rc2-mm1/kernel/fork.c
=====
--- 2.6.25-rc2-mm1.orig/kernel/fork.c
+++ 2.6.25-rc2-mm1/kernel/fork.c
@@ -549,7 +549,7 @@ fail_nocontext:
    return NULL;
}

-static int copy_mm(unsigned long clone_flags, struct task_struct * tsk)
+static int copy_mm(u64 clone_flags, struct task_struct *tsk)
{
    struct mm_struct * mm, *oldmm;
    int retval;
@@ -625,7 +625,7 @@ struct fs_struct *copy_fs_struct(struct

EXPORT_SYMBOL_GPL(copy_fs_struct);

-static int copy_fs(unsigned long clone_flags, struct task_struct *tsk)
+static int copy_fs(u64 clone_flags, struct task_struct *tsk)
{

```

```

if (clone_flags & CLONE_FS) {
    atomic_inc(&current->fs->count);
@@ -767,7 +767,7 @@ out:
    return NULL;
}

```

```

-static int copy_files(unsigned long clone_flags, struct task_struct * tsk)
+static int copy_files(u64 clone_flags, struct task_struct *tsk)
{
    struct files_struct *oldf, *newf;
    int error = 0;
@@ -800,7 +800,7 @@ out:
    return error;
}

```

```

-static int copy_io(unsigned long clone_flags, struct task_struct *tsk)
+static int copy_io(u64 clone_flags, struct task_struct *tsk)
{
#ifdef CONFIG_BLOCK
    struct io_context *ioc = current->io_context;
@@ -853,7 +853,7 @@ int unshare_files(void)

```

```
EXPORT_SYMBOL(unshare_files);
```

```

-static int copy_sighand(unsigned long clone_flags, struct task_struct *tsk)
+static int copy_sighand(u64 clone_flags, struct task_struct *tsk)
{
    struct sighand_struct *sig;

@@ -876,7 +876,7 @@ void __cleanup_sighand(struct sighand_st
    kmem_cache_free(sighand_cachep, sighand);
}

```

```

-static int copy_signal(unsigned long clone_flags, struct task_struct *tsk)
+static int copy_signal(u64 clone_flags, struct task_struct *tsk)
{
    struct signal_struct *sig;
    int ret;
@@ -967,7 +967,7 @@ static void cleanup_signal(struct task_s
    __cleanup_signal(sig);
}

```

```

-static void copy_flags(unsigned long clone_flags, struct task_struct *p)
+static void copy_flags(u64 clone_flags, struct task_struct *p)
{
    unsigned long new_flags = p->flags;

```

```
@@ -1003,7 +1003,7 @@ static void rt_mutex_init_task(struct ta
```

```

* parts of the process environment (as per the clone
* flags). The actual kick-off is left to the caller.
*/
-static struct task_struct *copy_process(unsigned long clone_flags,
+static struct task_struct *copy_process(u64 clone_flags,
    unsigned long stack_start,
    struct pt_regs *regs,
    unsigned long stack_size,
@@ -1425,7 +1425,7 @@ struct task_struct * __cpuinit fork_idle
    return task;
}

-static int fork_traceflag(unsigned clone_flags)
+static int fork_traceflag(u64 clone_flags)
{
    if (clone_flags & CLONE_UNTRACED)
        return 0;
@@ -1447,7 +1447,7 @@ static int fork_traceflag(unsigned clone
    * It copies the process, and if successful kick-starts
    * it and waits for it to finish using the VM if required.
*/
-long do_fork(unsigned long clone_flags,
+long do_fork(u64 clone_flags,
    unsigned long stack_start,
    struct pt_regs *regs,
    unsigned long stack_size,
@@ -1469,7 +1469,7 @@ long do_fork(unsigned long clone_flags,

    count--;
    printk(KERN_INFO "fork(): process '%s' used deprecated "
-   "clone flags 0x%lx\n",
+   "clone flags 0x%llx\n",
        get_task_comm(comm, current),
        clone_flags & CLONE_STOPPED);
}
@@ -1572,7 +1572,7 @@ void __init proc_caches_init(void)
    * Check constraints on flags passed to the unshare system call and
    * force unsharing of additional process context as appropriate.
*/
-static void check_unshare_flags(unsigned long *flags_ptr)
+static void check_unshare_flags(u64 *flags_ptr)
{
    /*
    * If unsharing a thread from a thread group, must also
@@ -1605,7 +1605,7 @@ static void check_unshare_flags(unsigned
    /*
    * Unsharing of tasks created with CLONE_THREAD is not supported yet
    */

```

```

-static int unshare_thread(unsigned long unshare_flags)
+static int unshare_thread(u64 unshare_flags)
{
    if (unshare_flags & CLONE_THREAD)
        return -EINVAL;
@@ -1616,7 +1616,7 @@ static int unshare_thread(unsigned long
/*
 * Unshare the filesystem structure if it is being shared
 */
-static int unshare_fs(unsigned long unshare_flags, struct fs_struct **new_fsp)
+static int unshare_fs(u64 unshare_flags, struct fs_struct **new_fsp)
{
    struct fs_struct *fs = current->fs;

@@ -1633,7 +1633,7 @@ static int unshare_fs(unsigned long unsh
/*
 * Unsharing of sighand is not supported yet
 */
-static int unshare_sighand(unsigned long unshare_flags, struct sighand_struct **new_sighp)
+static int unshare_sighand(u64 unshare_flags, struct sighand_struct **new_sighp)
{
    struct sighand_struct *sigh = current->sighand;

@@ -1646,7 +1646,7 @@ static int unshare_sighand(unsigned long
/*
 * Unshare vm if it is being shared
 */
-static int unshare_vm(unsigned long unshare_flags, struct mm_struct **new_mmp)
+static int unshare_vm(u64 unshare_flags, struct mm_struct **new_mmp)
{
    struct mm_struct *mm = current->mm;

@@ -1661,7 +1661,7 @@ static int unshare_vm(unsigned long unsh
/*
 * Unshare file descriptor table if it is being shared
 */
-static int unshare_fd(unsigned long unshare_flags, struct files_struct **new_fdp)
+static int unshare_fd(u64 unshare_flags, struct files_struct **new_fdp)
{
    struct files_struct *fd = current->files;
    int error = 0;
@@ -1680,7 +1680,7 @@ static int unshare_fd(unsigned long unsh
 * Unsharing of semundo for tasks created with CLONE_SYSVSEM is not
 * supported yet
 */
-static int unshare_semundo(unsigned long unshare_flags, struct sem_undo_list **new_ulistp)
+static int unshare_semundo(u64 unshare_flags, struct sem_undo_list **new_ulistp)
{

```

```
if (unshare_flags & CLONE_SYSVSEM)
```

```
return -EINVAL;
```

```
Index: 2.6.25-rc2-mm1/security/dummy.c
```

```
-----  
--- 2.6.25-rc2-mm1.orig/security/dummy.c
```

```
+++ 2.6.25-rc2-mm1/security/dummy.c
```

```
@@ -493,7 +493,7 @@ static int dummy_dentry_open (struct fil
```

```
return 0;
```

```
}
```

```
-static int dummy_task_create (unsigned long clone_flags)
```

```
+static int dummy_task_create(u64 clone_flags)
```

```
{
```

```
return 0;
```

```
}
```

```
Index: 2.6.25-rc2-mm1/security/keys/process_keys.c
```

```
-----  
--- 2.6.25-rc2-mm1.orig/security/keys/process_keys.c
```

```
+++ 2.6.25-rc2-mm1/security/keys/process_keys.c
```

```
@@ -278,7 +278,7 @@ int copy_thread_group_keys(struct task_s
```

```
/*
```

```
* copy the keys for fork
```

```
*/
```

```
-int copy_keys(unsigned long clone_flags, struct task_struct *tsk)
```

```
+int copy_keys(u64 clone_flags, struct task_struct *tsk)
```

```
{
```

```
key_check(tsk->thread_keyring);
```

```
key_check(tsk->request_key_auth);
```

```
Index: 2.6.25-rc2-mm1/security/security.c
```

```
-----  
--- 2.6.25-rc2-mm1.orig/security/security.c
```

```
+++ 2.6.25-rc2-mm1/security/security.c
```

```
@@ -580,7 +580,7 @@ int security_dentry_open(struct file *fi
```

```
return security_ops->dentry_open(file);
```

```
}
```

```
-int security_task_create(unsigned long clone_flags)
```

```
+int security_task_create(u64 clone_flags)
```

```
{
```

```
return security_ops->task_create(clone_flags);
```

```
}
```

```
Index: 2.6.25-rc2-mm1/security/selinux/hooks.c
```

```
-----  
--- 2.6.25-rc2-mm1.orig/security/selinux/hooks.c
```

```
+++ 2.6.25-rc2-mm1/security/selinux/hooks.c
```

```
@@ -3036,7 +3036,7 @@ static int selinux_dentry_open(struct fi
```

```
/* task security operations */
```



```
-static int selinux_task_create(unsigned long clone_flags)
+static int selinux_task_create(u64 clone_flags)
{
    int rc;
```

Index: 2.6.25-rc2-mm1/include/linux/nsproxy.h

```
=====
--- 2.6.25-rc2-mm1.orig/include/linux/nsproxy.h
+++ 2.6.25-rc2-mm1/include/linux/nsproxy.h
@@ -62,11 +62,11 @@ static inline struct nsproxy *task_nspro
    return rcu_dereference(tsk->nsproxy);
}
```

```
-int copy_namespaces(unsigned long flags, struct task_struct *tsk);
+int copy_namespaces(u64 clone_flags, struct task_struct *tsk);
void exit_task_namespaces(struct task_struct *tsk);
void switch_task_namespaces(struct task_struct *tsk, struct nsproxy *new);
void free_nsproxy(struct nsproxy *ns);
-int unshare_nsproxy_namespaces(unsigned long, struct nsproxy **,
+int unshare_nsproxy_namespaces(u64, struct nsproxy **,
    struct fs_struct *);
```

```
static inline void put_nsproxy(struct nsproxy *ns)
```

Index: 2.6.25-rc2-mm1/kernel/nsproxy.c

```
=====
--- 2.6.25-rc2-mm1.orig/kernel/nsproxy.c
+++ 2.6.25-rc2-mm1/kernel/nsproxy.c
@@ -47,7 +47,7 @@ static inline struct nsproxy *clone_nspr
    * Return the newly created nsproxy. Do not attach this to the task,
    * leave it to the caller to do proper locking and attach it to task.
    */
-static struct nsproxy *create_new_namespaces(unsigned long flags,
+static struct nsproxy *create_new_namespaces(u64 flags,
    struct task_struct *tsk, struct fs_struct *new_fs)
{
    struct nsproxy *new_nsp;
@@ -119,7 +119,7 @@ out_ns:
    * called from clone. This now handles copy for nsproxy and all
    * namespaces therein.
    */
-int copy_namespaces(unsigned long flags, struct task_struct *tsk)
+int copy_namespaces(u64 flags, struct task_struct *tsk)
{
    struct nsproxy *old_ns = tsk->nsproxy;
    struct nsproxy *new_ns;
@@ -178,7 +178,7 @@ void free_nsproxy(struct nsproxy *ns)
    * Called from unshare. Unshare all the namespaces part of nsproxy.
```

```

* On success, returns the new nsproxy.
*/
-int unshare_nsproxy_namespaces(unsigned long unshare_flags,
+int unshare_nsproxy_namespaces(u64 unshare_flags,
    struct nsproxy **new_nsp, struct fs_struct *new_fs)
{
    int err = 0;
Index: 2.6.25-rc2-mm1/fs/namespace.c
=====
--- 2.6.25-rc2-mm1.orig/fs/namespace.c
+++ 2.6.25-rc2-mm1/fs/namespace.c
@@ -1987,7 +1987,7 @@ static struct mnt_namespace *dup_mnt_ns(
    return new_ns;
}

-struct mnt_namespace *copy_mnt_ns(unsigned long flags, struct mnt_namespace *ns,
+struct mnt_namespace *copy_mnt_ns(u64 flags, struct mnt_namespace *ns,
    struct fs_struct *new_fs)
{
    struct mnt_namespace *new_ns;
Index: 2.6.25-rc2-mm1/include/linux/ipc_namespace.h
=====
--- 2.6.25-rc2-mm1.orig/include/linux/ipc_namespace.h
+++ 2.6.25-rc2-mm1/include/linux/ipc_namespace.h
@@ -62,7 +62,7 @@ extern int ipcns_notify(unsigned long);

#if defined(CONFIG_SYSVIPC) && defined(CONFIG_IPC_NS)
extern void free_ipc_ns(struct kref *kref);
-extern struct ipc_namespace *copy_ipcs(unsigned long flags,
+extern struct ipc_namespace *copy_ipcs(u64 flags,
    struct ipc_namespace *ns);
extern void free_ipcs(struct ipc_namespace *ns, struct ipc_ids *ids,
    void (*free)(struct ipc_namespace *),
@@ -80,7 +80,7 @@ static inline void put_ipc_ns(struct ipc
    kref_put(&ns->kref, free_ipc_ns);
}
#else
-static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
+static inline struct ipc_namespace *copy_ipcs(u64 flags,
    struct ipc_namespace *ns)
{
    if (flags & CLONE_NEWIPC)
Index: 2.6.25-rc2-mm1/include/linux/mnt_namespace.h
=====
--- 2.6.25-rc2-mm1.orig/include/linux/mnt_namespace.h
+++ 2.6.25-rc2-mm1/include/linux/mnt_namespace.h
@@ -14,7 +14,7 @@ struct mnt_namespace {
    int event;

```

```
};
```

```
-extern struct mnt_namespace *copy_mnt_ns(unsigned long, struct mnt_namespace *,
+extern struct mnt_namespace *copy_mnt_ns(u64, struct mnt_namespace *,
    struct fs_struct *);
extern void __put_mnt_ns(struct mnt_namespace *ns);
```

Index: 2.6.25-rc2-mm1/include/linux/pid\_namespace.h

```
=====
--- 2.6.25-rc2-mm1.orig/include/linux/pid_namespace.h
+++ 2.6.25-rc2-mm1/include/linux/pid_namespace.h
@@ -37,7 +37,7 @@ static inline struct pid_namespace *get_
    return ns;
}
```

```
-extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *ns);
+extern struct pid_namespace *copy_pid_ns(u64 flags, struct pid_namespace *ns);
extern void free_pid_ns(struct kref *kref);
extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
```

```
@@ -56,7 +56,7 @@ static inline struct pid_namespace *get_
}
```

```
static inline struct pid_namespace *
-copy_pid_ns(unsigned long flags, struct pid_namespace *ns)
+copy_pid_ns(u64 flags, struct pid_namespace *ns)
```

```
{
    if (flags & CLONE_NEWPID)
        ns = ERR_PTR(-EINVAL);
```

Index: 2.6.25-rc2-mm1/include/linux/user\_namespace.h

```
=====
--- 2.6.25-rc2-mm1.orig/include/linux/user_namespace.h
+++ 2.6.25-rc2-mm1/include/linux/user_namespace.h
@@ -26,7 +26,7 @@ static inline struct user_namespace *get_
    return ns;
}
```

```
-extern struct user_namespace *copy_user_ns(int flags,
+extern struct user_namespace *copy_user_ns(u64 flags,
    struct user_namespace *old_ns);
extern void free_user_ns(struct kref *kref);
```

```
@@ -43,7 +43,7 @@ static inline struct user_namespace *get_
    return &init_user_ns;
}
```

```
-static inline struct user_namespace *copy_user_ns(int flags,
+static inline struct user_namespace *copy_user_ns(u64 flags,
```

```

    struct user_namespace *old_ns)
{
    if (flags & CLONE_NEWUSER)
Index: 2.6.25-rc2-mm1/include/linux/utsname.h
=====
--- 2.6.25-rc2-mm1.orig/include/linux/utsname.h
+++ 2.6.25-rc2-mm1/include/linux/utsname.h
@@ -50,7 +50,7 @@ static inline void get_uts_ns(struct uts
    kref_get(&ns->kref);
}

-extern struct uts_namespace *copy_utsname(unsigned long flags,
+extern struct uts_namespace *copy_utsname(u64 flags,
    struct uts_namespace *ns);
extern void free_uts_ns(struct kref *kref);

@@ -67,7 +67,7 @@ static inline void put_uts_ns(struct uts
{
}

-static inline struct uts_namespace *copy_utsname(unsigned long flags,
+static inline struct uts_namespace *copy_utsname(u64 flags,
    struct uts_namespace *ns)
{
    if (flags & CLONE_NEWUTS)
Index: 2.6.25-rc2-mm1/include/net/net_namespace.h
=====
--- 2.6.25-rc2-mm1.orig/include/net/net_namespace.h
+++ 2.6.25-rc2-mm1/include/net/net_namespace.h
@@ -73,9 +73,9 @@ extern struct net init_net;
extern struct list_head net_namespace_list;

#ifdef CONFIG_NET
-extern struct net *copy_net_ns(unsigned long flags, struct net *net_ns);
+extern struct net *copy_net_ns(u64 flags, struct net *net_ns);
#else
-static inline struct net *copy_net_ns(unsigned long flags, struct net *net_ns)
+static inline struct net *copy_net_ns(u64 flags, struct net *net_ns)
{
    /* There is nothing to copy so this is a noop */
    return net_ns;
Index: 2.6.25-rc2-mm1/ipc/namespace.c
=====
--- 2.6.25-rc2-mm1.orig/ipc/namespace.c
+++ 2.6.25-rc2-mm1/ipc/namespace.c
@@ -38,7 +38,7 @@ static struct ipc_namespace *clone_ipc_n
    return ns;
}

```

```
-struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
+struct ipc_namespace *copy_ipcs(u64 flags, struct ipc_namespace *ns)
{
    struct ipc_namespace *new_ns;
```

Index: 2.6.25-rc2-mm1/kernel/pid\_namespace.c

```
=====
--- 2.6.25-rc2-mm1.orig/kernel/pid_namespace.c
+++ 2.6.25-rc2-mm1/kernel/pid_namespace.c
@@ -115,7 +115,7 @@ static void destroy_pid_namespace(struct
    kmem_cache_free(pid_ns_cachep, ns);
}
```

```
-struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *old_ns)
+struct pid_namespace *copy_pid_ns(u64 flags, struct pid_namespace *old_ns)
{
    struct pid_namespace *new_ns;
```

Index: 2.6.25-rc2-mm1/kernel/user\_namespace.c

```
=====
--- 2.6.25-rc2-mm1.orig/kernel/user_namespace.c
+++ 2.6.25-rc2-mm1/kernel/user_namespace.c
@@ -49,7 +49,7 @@ static struct user_namespace *clone_user
    return ns;
}
```

```
-struct user_namespace *copy_user_ns(int flags, struct user_namespace *old_ns)
+struct user_namespace *copy_user_ns(u64 flags, struct user_namespace *old_ns)
{
    struct user_namespace *new_ns;
```

Index: 2.6.25-rc2-mm1/kernel/utsname.c

```
=====
--- 2.6.25-rc2-mm1.orig/kernel/utsname.c
+++ 2.6.25-rc2-mm1/kernel/utsname.c
@@ -41,7 +41,7 @@ static struct uts_namespace *clone_uts_n
    * utsname of this process won't be seen by parent, and vice
    * versa.
    */
-struct uts_namespace *copy_utsname(unsigned long flags, struct uts_namespace *old_ns)
+struct uts_namespace *copy_utsname(u64 flags, struct uts_namespace *old_ns)
{
    struct uts_namespace *new_ns;
```

Index: 2.6.25-rc2-mm1/net/core/net\_namespace.c

```
=====
--- 2.6.25-rc2-mm1.orig/net/core/net_namespace.c
```

```

+++ 2.6.25-rc2-mm1/net/core/net_namespace.c
@@ -79,7 +79,7 @@ static void net_free(struct net *net)
    kmem_cache_free(net_cachep, net);
}

-struct net *copy_net_ns(unsigned long flags, struct net *old_net)
+struct net *copy_net_ns(u64 flags, struct net *old_net)
{
    struct net *new_net = NULL;
    int err;
@@ -155,7 +155,7 @@ void __put_net(struct net *net)
EXPORT_SYMBOL_GPL(__put_net);

#else
-struct net *copy_net_ns(unsigned long flags, struct net *old_net)
+struct net *copy_net_ns(u64 flags, struct net *old_net)
{
    if (flags & CLONE_NEWNET)
        return ERR_PTR(-EINVAL);
--

```

-----

You received this message because you are subscribed to the Google Groups "lxc-dev" group.  
 To post to this group, send email to [lxc-dev@googlegroups.com](mailto:lxc-dev@googlegroups.com)  
 To unsubscribe from this group, send email to [lxc-dev-unsubscribe@googlegroups.com](mailto:lxc-dev-unsubscribe@googlegroups.com)  
 For more options, visit this group at <http://groups.google.com/group/lxc-dev?hl=en>

-----

---

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

---



---

Subject: [PATCH 2/3] add do\_unshare()  
 Posted by [Sukadev Bhattiprolu](#) on Wed, 09 Apr 2008 22:34:06 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

From: Sukadev Bhattiprolu <[sukadev@us.ibm.com](mailto:sukadev@us.ibm.com)>  
 Subject: [PATCH 2/3] add do\_unshare()

This patch adds a do\_unshare() routine which will be common to the unshare() and unshare64() syscall.

Signed-off-by: Cedric Le Goater <[clg@fr.ibm.com](mailto:clg@fr.ibm.com)>  
 Signed-off-by: Sukadev Bhattiprolu <[sukadev@us.ibm.com](mailto:sukadev@us.ibm.com)>

---

kernel/fork.c | 7 ++++++-  
1 file changed, 6 insertions(+), 1 deletion(-)

Index: 2.6.25-rc2-mm1/kernel/fork.c

```
=====
--- 2.6.25-rc2-mm1.orig/kernel/fork.c
+++ 2.6.25-rc2-mm1/kernel/fork.c
@@ -1696,7 +1696,7 @@ static int unshare_semundo(u64 unshare_f
 * constructed. Here we are modifying the current, active,
 * task_struct.
 */
-asmlinkage long sys_unshare(unsigned long unshare_flags)
+static long do_unshare(u64 unshare_flags)
{
    int err = 0;
    struct fs_struct *fs, *new_fs = NULL;
@@ -1790,3 +1790,8 @@ bad_unshare_cleanup_thread:
bad_unshare_out:
    return err;
}
+
+asmlinkage long sys_unshare(unsigned long unshare_flags)
+{
+ return do_unshare(unshare_flags);
+}
--
```

~~~~~  
You received this message because you are subscribed to the Google Groups "lxc-dev" group.  
To post to this group, send email to [lxc-dev@googlegroups.com](mailto:lxc-dev@googlegroups.com)  
To unsubscribe from this group, send email to [lxc-dev-unsubscribe@googlegroups.com](mailto:lxc-dev-unsubscribe@googlegroups.com)  
For more options, visit this group at <http://groups.google.com/group/lxc-dev?hl=en>  
~~~~~

---

Containers mailing list  
[Containers@lists.linux-foundation.org](mailto:Containers@lists.linux-foundation.org)  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: [PATCH 3/3] add the clone64() and unshare64() syscalls  
Posted by [Sukadev Bhattiprolu](#) on Wed, 09 Apr 2008 22:34:59 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

From: Cedric Le Goater <[clg@fr.ibm.com](mailto:clg@fr.ibm.com)>  
Subject: [PATCH 3/3] add the clone64() and unshare64() syscalls

This patch adds 2 new syscalls :

```
long sys_clone64(unsigned long flags_high, unsigned long flags_low,
unsigned long newsp);
```

```
long sys_unshare64(unsigned long flags_high, unsigned long flags_low);
```

The current version of clone64() does not support CLONE\_PARENT\_SETTID and CLONE\_CHILD\_CLEARTID because we would exceed the 6 registers limit of some arches. It's possible to get around this limitation but we might not need it as we already have clone()

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

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

---

```
arch/powerpc/kernel/entry_32.S | 8 ++++++++
arch/powerpc/kernel/entry_64.S | 5 +++++
arch/powerpc/kernel/process.c | 15 ++++++
arch/s390/kernel/compat_linux.c | 16 ++++++
arch/s390/kernel/compat_wrapper.S | 6 +++++
arch/s390/kernel/process.c | 15 ++++++
arch/s390/kernel/syscalls.S | 2 ++
arch/x86/ia32/ia32entry.S | 4 ++++
arch/x86/ia32/sys_ia32.c | 12 ++++++
arch/x86/kernel/entry_64.S | 1 +
arch/x86/kernel/process_32.c | 14 ++++++
arch/x86/kernel/process_64.c | 15 ++++++
arch/x86/kernel/syscall_table_32.S | 2 ++
include/asm-powerpc/systbl.h | 2 ++
include/asm-powerpc/unistd.h | 4 +++-
include/asm-s390/unistd.h | 4 +++-
include/asm-x86/unistd_32.h | 2 ++
include/asm-x86/unistd_64.h | 4 ++++
include/linux/syscalls.h | 3 +++
kernel/fork.c | 7 ++++++
kernel/sys_ni.c | 3 +++
21 files changed, 142 insertions(+), 2 deletions(-)
```

Index: 2.6.25-rc2-mm1/arch/s390/kernel/syscalls.S

```
=====
--- 2.6.25-rc2-mm1.orig/arch/s390/kernel/syscalls.S 2008-02-27 15:17:34.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/s390/kernel/syscalls.S 2008-03-06 22:08:49.000000000 -0800
@@ -330,3 +330,5 @@ SYSCALL(sys_eventfd,sys_eventfd,sys_even
SYSCALL(sys_timerfd_create,sys_timerfd_create,sys_timerfd_create_wrapper)
SYSCALL(sys_timerfd_settime,sys_timerfd_settime,compat_sys_timerfd_settime_wrapper) /*
320 */
SYSCALL(sys_timerfd_gettime,sys_timerfd_gettime,compat_sys_timerfd_gettime_wrapper)
```



```

+SYSCALL(sys_clone64,sys_clone64,sys32_clone64)
+SYSCALL(sys_unshare64,sys_unshare64,sys_unshare64_wrapper)
Index: 2.6.25-rc2-mm1/arch/x86/kernel/syscall_table_32.S
=====
--- 2.6.25-rc2-mm1.orig/arch/x86/kernel/syscall_table_32.S 2008-02-27 15:17:35.000000000
-0800
+++ 2.6.25-rc2-mm1/arch/x86/kernel/syscall_table_32.S 2008-03-06 22:08:49.000000000 -0800
@@ -326,3 +326,5 @@ ENTRY(sys_call_table)
 .long sys_fallocate
 .long sys_timerfd_settime /* 325 */
 .long sys_timerfd_gettime
+ .long sys_clone64
+ .long sys_unshare64
Index: 2.6.25-rc2-mm1/include/asm-powerpc/systbl.h
=====
--- 2.6.25-rc2-mm1.orig/include/asm-powerpc/systbl.h 2008-02-27 15:18:12.000000000 -0800
+++ 2.6.25-rc2-mm1/include/asm-powerpc/systbl.h 2008-03-06 22:08:49.000000000 -0800
@@ -316,3 +316,5 @@ COMPAT_SYS(fallocate)
 SYSCALL(subpage_prot)
 COMPAT_SYS_SPU(timerfd_settime)
 COMPAT_SYS_SPU(timerfd_gettime)
+PPC_SYS(clone64)
+SYSCALL_SPU(unshare64)
Index: 2.6.25-rc2-mm1/include/asm-powerpc/unistd.h
=====
--- 2.6.25-rc2-mm1.orig/include/asm-powerpc/unistd.h 2008-02-27 15:18:12.000000000 -0800
+++ 2.6.25-rc2-mm1/include/asm-powerpc/unistd.h 2008-03-06 22:08:49.000000000 -0800
@@ -335,10 +335,12 @@
#define __NR_subpage_prot 310
#define __NR_timerfd_settime 311
#define __NR_timerfd_gettime 312
+#define __NR_clone64 313
+#define __NR_unshare64 314

#ifdef __KERNEL__

-#define __NR_syscalls 313
+#define __NR_syscalls 315

#define __NR__exit __NR_exit
#define NR_syscalls __NR_syscalls
Index: 2.6.25-rc2-mm1/include/asm-s390/unistd.h
=====
--- 2.6.25-rc2-mm1.orig/include/asm-s390/unistd.h 2008-02-27 15:18:13.000000000 -0800
+++ 2.6.25-rc2-mm1/include/asm-s390/unistd.h 2008-03-06 22:08:49.000000000 -0800
@@ -259,7 +259,9 @@
#define __NR_timerfd_create 319
#define __NR_timerfd_settime 320

```

```

#define __NR_timerfd_gettime 321
-#define NR_syscalls 322
+#define __NR_clone64 322
+#define __NR_unshare64 323
+#define NR_syscalls 324

/*
 * There are some system calls that are not present on 64 bit, some
Index: 2.6.25-rc2-mm1/include/asm-x86/unistd_32.h
=====
--- 2.6.25-rc2-mm1.orig/include/asm-x86/unistd_32.h 2008-02-27 15:18:16.000000000 -0800
+++ 2.6.25-rc2-mm1/include/asm-x86/unistd_32.h 2008-03-06 22:08:49.000000000 -0800
@@ -332,6 +332,8 @@
#define __NR_fallocate 324
#define __NR_timerfd_settime 325
#define __NR_timerfd_gettime 326
+#define __NR_clone64 327
+#define __NR_unshare64 328

#ifdef __KERNEL__

Index: 2.6.25-rc2-mm1/include/asm-x86/unistd_64.h
=====
--- 2.6.25-rc2-mm1.orig/include/asm-x86/unistd_64.h 2008-02-27 15:18:16.000000000 -0800
+++ 2.6.25-rc2-mm1/include/asm-x86/unistd_64.h 2008-03-06 22:08:49.000000000 -0800
@@ -639,6 +639,10 @@ __SYSCALL(__NR_fallocate, sys_fallocate)
__SYSCALL(__NR_timerfd_settime, sys_timerfd_settime)
#define __NR_timerfd_gettime 287
__SYSCALL(__NR_timerfd_gettime, sys_timerfd_gettime)
+#define __NR_clone64 288
+__SYSCALL(__NR_clone64, stub_clone64)
+#define __NR_unshare64 289
+__SYSCALL(__NR_unshare64, sys_unshare64)

#ifdef __NO_STUBS
Index: 2.6.25-rc2-mm1/include/linux/syscalls.h
=====
--- 2.6.25-rc2-mm1.orig/include/linux/syscalls.h 2008-02-27 15:18:18.000000000 -0800
+++ 2.6.25-rc2-mm1/include/linux/syscalls.h 2008-03-06 22:08:49.000000000 -0800
@@ -615,6 +615,9 @@ asmlinkage long sys_timerfd_gettime(int
asmlinkage long sys_eventfd(unsigned int count);
asmlinkage long sys_fallocate(int fd, int mode, loff_t offset, loff_t len);

+asmlinkage long sys_unshare64(unsigned long clone_flags_high,
+ unsigned long clone_flags_low);
+
int kernel_execve(const char *filename, char *const argv[], char *const envp[]);

```

#endif

Index: 2.6.25-rc2-mm1/kernel/sys\_ni.c

=====  
--- 2.6.25-rc2-mm1.orig/kernel/sys\_ni.c 2008-02-27 15:18:23.000000000 -0800

+++ 2.6.25-rc2-mm1/kernel/sys\_ni.c 2008-03-06 22:08:49.000000000 -0800

@@ -161,3 +161,6 @@ cond\_syscall(sys\_timerfd\_gettime);

cond\_syscall(compat\_sys\_timerfd\_settime);

cond\_syscall(compat\_sys\_timerfd\_gettime);

cond\_syscall(sys\_eventfd);

+

+cond\_syscall(sys\_clone64);

+cond\_syscall(sys\_unshare64);

Index: 2.6.25-rc2-mm1/arch/x86/kernel/process\_32.c

=====  
--- 2.6.25-rc2-mm1.orig/arch/x86/kernel/process\_32.c 2008-03-06 22:08:49.000000000 -0800

+++ 2.6.25-rc2-mm1/arch/x86/kernel/process\_32.c 2008-03-06 22:08:49.000000000 -0800

@@ -771,6 +771,20 @@ asmlinkage int sys\_clone(struct pt\_regs

return do\_fork(clone\_flags, newsp, &regs, 0, parent\_tidptr, child\_tidptr);

}

+asmlinkage int sys\_clone64(struct pt\_regs regs)

{

+ u64 clone\_flags;

+ unsigned long newsp;

+

+ clone\_flags = ((u64) regs.bx << 32 | regs.cx);

+ clone\_flags &= ~(CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID);

+

+ newsp = regs.dx;

+ if (!newsp)

+ newsp = regs.sp;

+ return do\_fork(clone\_flags, newsp, &regs, 0, NULL, NULL);

+

+

/\*

\* This is trivial, and on the face of it looks like it

\* could equally well be done in user mode.

Index: 2.6.25-rc2-mm1/arch/x86/kernel/process\_64.c

=====  
--- 2.6.25-rc2-mm1.orig/arch/x86/kernel/process\_64.c 2008-03-06 22:08:49.000000000 -0800

+++ 2.6.25-rc2-mm1/arch/x86/kernel/process\_64.c 2008-03-06 22:08:49.000000000 -0800

@@ -775,6 +775,21 @@ sys\_clone(unsigned long clone\_flags, uns

return do\_fork(clone\_flags, newsp, regs, 0, parent\_tid, child\_tid);

}

+asmlinkage long

+sys\_clone64(unsigned long clone\_flags\_high, unsigned long clone\_flags\_low,

```

+ unsigned long newsp, struct pt_regs *regs)
+{
+ u64 clone_flags;
+
+ clone_flags = ((u64) clone_flags_high << 32 | clone_flags_low);
+ clone_flags &= ~(CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID);
+
+ if (!newsp)
+ newsp = regs->sp;
+ return do_fork(clone_flags, newsp, regs, 0, NULL, NULL);
+}

```

```

+
+
+/*
+ * This is trivial, and on the face of it looks like it
+ * could equally well be done in user mode.

```

Index: 2.6.25-rc2-mm1/arch/s390/kernel/compat\_linux.c

```

=====
--- 2.6.25-rc2-mm1.orig/arch/s390/kernel/compat_linux.c 2008-01-26 09:48:58.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/s390/kernel/compat_linux.c 2008-03-06 22:08:49.000000000 -0800
@@ -940,6 +940,22 @@ asmlinkage long sys32_clone(void)
     parent_tidptr, child_tidptr);
}

```

```

+asmlinkage long sys32_clone64(void)
+{
+ struct pt_regs *regs = task_pt_regs(current);
+ u64 clone_flags;
+ unsigned long newsp;
+
+ clone_flags = ((u64) (regs->orig_gpr2 & 0xffffffffUL) << 32 |
+ (regs->gprs[3] & 0xffffffffUL));
+ clone_flags &= ~(CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID);
+
+ newsp = regs->gprs[4] & 0x7fffffffUL;
+ if (!newsp)
+ newsp = regs->gprs[15];
+ return do_fork(clone_flags, newsp, regs, 0, NULL, NULL);
+}

```

```

+/*
+ * 31 bit emulation wrapper functions for sys_fadvise64/fadvise64_64.
+ * These need to rewrite the advise values for POSIX_FADV_{DONTNEED,NOREUSE}

```

Index: 2.6.25-rc2-mm1/arch/s390/kernel/process.c

```

=====
--- 2.6.25-rc2-mm1.orig/arch/s390/kernel/process.c 2008-03-06 22:08:49.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/s390/kernel/process.c 2008-03-06 22:08:49.000000000 -0800
@@ -325,6 +325,21 @@ asmlinkage long sys_clone(void)

```

```

    parent_tidptr, child_tidptr);
}

+asmlinkage long sys_clone64(void)
+{
+ struct pt_regs *regs = task_pt_regs(current);
+ u64 clone_flags;
+ unsigned long newsp;
+
+ clone_flags = ((u64) regs->orig_gpr2 << 32 | regs->gprs[3]);
+ clone_flags &= ~(CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID);
+
+ newsp = regs->gprs[4];
+ if (!newsp)
+ newsp = regs->gprs[15];
+ return do_fork(clone_flags, newsp, regs, 0, NULL, NULL);
+}
+
+/*

```

```

* This is trivial, and on the face of it looks like it
* could equally well be done in user mode.

```

```

Index: 2.6.25-rc2-mm1/arch/powerpc/kernel/process.c

```

```

=====
--- 2.6.25-rc2-mm1.orig/arch/powerpc/kernel/process.c 2008-03-06 22:08:49.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/powerpc/kernel/process.c 2008-03-06 22:08:49.000000000 -0800
@@ -829,6 +829,21 @@ int sys_clone(unsigned long clone_flags,
    return do_fork(clone_flags, usp, regs, 0, parent_tidp, child_tidp);
}

```

```

+int sys_clone64(unsigned long clone_flags_high, unsigned long clone_flags_low,
+ unsigned long usp, unsigned long p4, unsigned long p5,
+ unsigned long p6, struct pt_regs *regs)
+{
+ u64 clone_flags;
+
+ clone_flags = ((u64) clone_flags_high << 32 | clone_flags_low);
+ clone_flags &= ~(CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID);
+
+ CHECK_FULL_REGS(regs);
+ if (usp == 0)
+ usp = regs->gpr[1]; /* stack pointer for child */
+ return do_fork(clone_flags, usp, regs, 0, NULL, NULL);
+}
+
+int sys_fork(unsigned long p1, unsigned long p2, unsigned long p3,
+ unsigned long p4, unsigned long p5, unsigned long p6,
+ struct pt_regs *regs)

```

```

Index: 2.6.25-rc2-mm1/arch/x86/kernel/entry_64.S

```

```

=====
--- 2.6.25-rc2-mm1.orig/arch/x86/kernel/entry_64.S 2008-02-27 16:07:43.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/x86/kernel/entry_64.S 2008-03-06 22:08:49.000000000 -0800
@@ -527,6 +527,7 @@ END(label)
PTREGSCALL stub_rt_sigsuspend, sys_rt_sigsuspend, %rdx
PTREGSCALL stub_sigaltstack, sys_sigaltstack, %rdx
PTREGSCALL stub_iopl, sys_iopl, %rsi
+ PTREGSCALL stub_clone64, sys_clone64, %rcx

ENTRY(ptregscall_common)
popq %r11
Index: 2.6.25-rc2-mm1/arch/powerpc/kernel/entry_32.S
=====
--- 2.6.25-rc2-mm1.orig/arch/powerpc/kernel/entry_32.S 2008-01-26 09:48:57.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/powerpc/kernel/entry_32.S 2008-03-06 22:08:49.000000000 -0800
@@ -452,6 +452,14 @@ ppc_clone:
stw r0,_TRAP(r1) /* register set saved */
b sys_clone

+ .globl ppc_clone64
+ppc_clone64:
+ SAVE_NVGPRS(r1)
+ lwz r0,_TRAP(r1)
+ rlwinm r0,r0,0,0,30 /* clear LSB to indicate full */
+ stw r0,_TRAP(r1) /* register set saved */
+ b sys_clone64
+
.globl ppc_swapcontext
ppc_swapcontext:
SAVE_NVGPRS(r1)
Index: 2.6.25-rc2-mm1/arch/powerpc/kernel/entry_64.S
=====
--- 2.6.25-rc2-mm1.orig/arch/powerpc/kernel/entry_64.S 2008-01-26 09:48:57.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/powerpc/kernel/entry_64.S 2008-03-06 22:08:49.000000000 -0800
@@ -298,6 +298,11 @@ _GLOBAL(ppc_clone)
bl .sys_clone
b syscall_exit

+_GLOBAL(ppc_clone64)
+ bl .save_nvgprs
+ bl .sys_clone64
+ b syscall_exit
+
_GLOBAL(ppc32_swapcontext)
bl .save_nvgprs
bl .compat_sys_swapcontext
Index: 2.6.25-rc2-mm1/arch/s390/kernel/compat_wrapper.S
=====

```

```

--- 2.6.25-rc2-mm1.orig/arch/s390/kernel/compat_wrapper.S 2008-02-27 15:17:33.000000000
-0800
+++ 2.6.25-rc2-mm1/arch/s390/kernel/compat_wrapper.S 2008-03-06 22:08:49.000000000 -0800
@@ -1732,3 +1732,9 @@ compat_sys_timerfd_gettime_wrapper:
    lgfr %r2,%r2    # int
    llgr %r3,%r3    # struct compat_itimerspec *
    jg compat_sys_timerfd_gettime
+
+ .globl sys_unshare64_wrapper
+sys_unshare64_wrapper:
+ llgr %r2,%r2    # unsigned long
+ llgr %r3,%r3    # unsigned long
+ jg sys_unshare64
Index: 2.6.25-rc2-mm1/kernel/fork.c
=====
--- 2.6.25-rc2-mm1.orig/kernel/fork.c 2008-03-06 22:08:49.000000000 -0800
+++ 2.6.25-rc2-mm1/kernel/fork.c 2008-03-10 20:47:10.000000000 -0700
@@ -1795,3 +1795,10 @@ asmlinkage long sys_unshare(unsigned lon
 {
    return do_unshare(unshare_flags);
 }
+
+asmlinkage long sys_unshare64(unsigned long flags_high, unsigned long flags_low)
+{
+ u64 unshare_flags = ((u64) flags_high << 32 | flags_low);
+
+ return do_unshare(unshare_flags);
+}
Index: 2.6.25-rc2-mm1/arch/x86/ia32/sys_ia32.c
=====
--- 2.6.25-rc2-mm1.orig/arch/x86/ia32/sys_ia32.c 2008-02-27 15:17:35.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/x86/ia32/sys_ia32.c 2008-03-06 22:08:49.000000000 -0800
@@ -824,6 +824,18 @@ asmlinkage long sys32_clone(unsigned int
    return do_fork(clone_flags, newsp, regs, 0, parent_tid, child_tid);
 }

+asmlinkage long sys32_clone64(unsigned int flags_high, unsigned int flags_low,
+ unsigned int newsp, struct pt_regs *regs)
+{
+ u64 clone_flags = ((u64) flags_high << 32 | flags_low);
+
+ clone_flags &= ~(CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID);
+
+ if (!newsp)
+   newsp = regs->sp;
+ return do_fork(clone_flags, newsp, regs, 0, NULL, NULL);
+}
+

```

```
/*
 * Some system calls that need sign extended arguments. This could be
 * done by a generic wrapper.
```

Index: 2.6.25-rc2-mm1/arch/x86/ia32/ia32entry.S

```
=====
--- 2.6.25-rc2-mm1.orig/arch/x86/ia32/ia32entry.S 2008-02-27 15:17:35.000000000 -0800
+++ 2.6.25-rc2-mm1/arch/x86/ia32/ia32entry.S 2008-03-06 22:08:49.000000000 -0800
@@ -373,6 +373,7 @@ quiet_ni_syscall:
    PTREGSCALL stub32_vfork, sys_vfork, %rdi
    PTREGSCALL stub32_iopl, sys_iopl, %rsi
    PTREGSCALL stub32_rt_sigsuspend, sys_rt_sigsuspend, %rdx
+ PTREGSCALL stub32_clone64, sys32_clone64, %rcx

ENTRY(ia32_ptregs_common)
    popq %r11
@@ -727,4 +728,7 @@ ia32_sys_call_table:
    .quad sys32_fallocate
    .quad compat_sys_timerfd_settime /* 325 */
    .quad compat_sys_timerfd_gettime
+ .quad stub32_clone64
+ .quad sys_unshare64
+
ia32_syscall_end:
```

---

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

---

---

Subject: Re: [PATCH 3/3] add the clone64() and unshare64() syscalls  
Posted by [Jakub Jelinek](#) on Wed, 09 Apr 2008 23:07:34 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, Apr 09, 2008 at 03:34:59PM -0700, sukadev@us.ibm.com wrote:  
> From: Cedric Le Goater <clg@fr.ibm.com>  
> Subject: [PATCH 3/3] add the clone64() and unshare64() syscalls  
>  
> This patch adds 2 new syscalls :  
>  
> long sys\_clone64(unsigned long flags\_high, unsigned long flags\_low,  
> unsigned long newsp);  
>  
> long sys\_unshare64(unsigned long flags\_high, unsigned long flags\_low);

Can you explain why are you adding it for 64-bit arches too? unsigned long is there already 64-bit, and both sys\_clone and sys\_unshare have unsigned long flags, rather than unsigned int.



Jakub

---

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

---

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [hpa](#) on Thu, 10 Apr 2008 00:00:25 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

sukadev@us.ibm.com wrote:

> This is a resend of the patch set Cedric had sent earlier. I ported  
> the patch set to 2.6.25-rc8-mm1 and tested on x86 and x86\_64.  
> ---  
>  
> We have run out of the 32 bits in clone\_flags !  
>  
> This patchset introduces 2 new system calls which support 64bit clone-flags.  
>  
> long sys\_clone64(unsigned long flags\_high, unsigned long flags\_low,  
> unsigned long newsp);  
>  
> long sys\_unshare64(unsigned long flags\_high, unsigned long flags\_low);  
>  
> The current version of clone64() does not support CLONE\_PARENT\_SETTID and  
> CLONE\_CHILD\_CLEARPID because we would exceed the 6 registers limit of some  
> arches. It's possible to get around this limitation but we might not  
> need it as we already have clone()  
>

I really dislike this interface.

If you're going to make it a 64-bit pass it in as a 64-bit number,  
instead of breaking it into two numbers. Better yet, IMO, would be to  
pass a pointer to a structure like:

```
struct shared {  
    unsigned long nwords;  
    unsigned long flags[];  
};
```

... which can be expanded indefinitely.

-hpa

---

Containers mailing list  
Containers@lists.linux-foundation.org

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [Sukadev Bhattiprolu](#) on Thu, 10 Apr 2008 01:07:17 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

H. Peter Anvin [[hpa@zytor.com](mailto:hpa@zytor.com)] wrote:

> sukadev@us.ibm.com wrote:

>> This is a resend of the patch set Cedric had sent earlier. I ported

>> the patch set to 2.6.25-rc8-mm1 and tested on x86 and x86\_64.

>> ---

>> We have run out of the 32 bits in clone\_flags !

>> This patchset introduces 2 new system calls which support 64bit

>> clone-flags.

>> long sys\_clone64(unsigned long flags\_high, unsigned long flags\_low,

>> unsigned long newsp);

>> long sys\_unshare64(unsigned long flags\_high, unsigned long

>> flags\_low);

>> The current version of clone64() does not support CLONE\_PARENT\_SETTID and

>> CLONE\_CHILD\_CLEARPID because we would exceed the 6 registers limit of some

>> arches. It's possible to get around this limitation but we might not

>> need it as we already have clone()

>

> I really dislike this interface.

>

> If you're going to make it a 64-bit pass it in as a 64-bit number, instead

> of breaking it into two numbers.

Maybe I am missing your point. The glibc interface could take a 64bit parameter, but don't we need to pass 32-bit values into the system call on 32 bit systems ?

> Better yet, IMO, would be to pass a pointer to a structure like:

>

> struct shared {

> unsigned long nwords;

> unsigned long flags[];

> };

>

> ... which can be expanded indefinitely.

Yes, this was discussed before in the context of Pavel Emelyanov's patch

<http://lkml.org/lkml/2008/1/16/109>

along with sys\_indirect(). While there was no consensus, it looked like adding a new system call was better than open ended interfaces.

---

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

---

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [hpa](#) on Thu, 10 Apr 2008 01:10:57 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

sukadev@us.ibm.com wrote:

>>  
>> If you're going to make it a 64-bit pass it in as a 64-bit number, instead  
>> of breaking it into two numbers.  
>  
> Maybe I am missing your point. The glibc interface could take a 64bit  
> parameter, but don't we need to pass 32-bit values into the system call  
> on 32 bit systems ?

Not as such, no. The ABI handles that. To make the ABI clean on some architectures, it's good to consider a 64-bit value only in positions where they map to an even:odd register pair once slotted in.

> Yes, this was discussed before in the context of Pavel Emelyanov's patch  
>  
> <http://lkml.org/lkml/2008/1/16/109>  
>  
> along with sys\_indirect(). While there was no consensus, it looked like  
> adding a new system call was better than open ended interfaces.

That's not really an open-ended interface, it's just an expandable bitmap.

-hpa

---

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

---

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [Sukadev Bhattiprolu](#) on Thu, 10 Apr 2008 02:38:18 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

H. Peter Anvin [[hpa@zytor.com](mailto:hpa@zytor.com)] wrote:

>> Yes, this was discussed before in the context of Pavel Emelyanov's patch  
>> <http://lkml.org/lkml/2008/1/16/109>  
>> along with sys\_indirect(). While there was no consensus, it looked like

>> adding a new system call was better than open ended interfaces.  
>  
> That's not really an open-ended interface, it's just an expandable bitmap.

Yes, we liked such an approach earlier too and its conceivable that we will run out of the 64-bits too :-)

But as Jon Corbet pointed out in the the thread above, it looked like adding a new system call has been the "traditional" way of solving this in Linux so far and there has been no consensus on a newer approach.

Sukadev

---

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

---

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [Paul Menage](#) on Thu, 10 Apr 2008 02:43:30 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, Apr 9, 2008 at 7:38 PM, <sukadev@us.ibm.com> wrote:

>  
> But as Jon Corbet pointed out in the the thread above, it looked like  
> adding a new system call has been the "traditional" way of solving this  
> in Linux so far and there has been no consensus on a newer approach.  
>

I thought that the consensus was that adding a new system call was better than trying to force extensibility on to the existing non-extensible system call.

But if we are adding a new system call, why not make the new one extensible to reduce the need for yet another new call in the future?

Paul

---

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

---

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [Cedric Le Goater](#) on Thu, 10 Apr 2008 06:48:50 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

H. Peter Anvin wrote:

> sukadev@us.ibm.com wrote:

>> This is a resend of the patch set Cedric had sent earlier. I ported

>> the patch set to 2.6.25-rc8-mm1 and tested on x86 and x86\_64.

>> ---

>>

>> We have run out of the 32 bits in clone\_flags !

>>

>> This patchset introduces 2 new system calls which support 64bit

>> clone-flags.

>>

>> long sys\_clone64(unsigned long flags\_high, unsigned long flags\_low,

>> unsigned long newsp);

>>

>> long sys\_unshare64(unsigned long flags\_high, unsigned long

>> flags\_low);

>>

>> The current version of clone64() does not support CLONE\_PARENT\_SETTID

>> and CLONE\_CHILD\_CLEARPID because we would exceed the 6 registers limit

>> of some arches. It's possible to get around this limitation but we

>> might not

>> need it as we already have clone()

>>

>

> I really dislike this interface.

>

> If you're going to make it a 64-bit pass it in as a 64-bit number,

> instead of breaking it into two numbers. Better yet, IMO, would be to

> pass a pointer to a structure like:

>

> struct shared {

> unsigned long nwords;

> unsigned long flags[];

> };

>

> ... which can be expanded indefinitely.

ok.

What about the copy\_from\_user() overhead ? is this something we care about for a clone like syscall ?

If not, this would certainly make our life simpler to extend clone flags.

I'm ready to implement anything if someone would just tell me in which direction.

Thanks !

C.

---

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

---

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Andi Kleen](#) on Thu, 10 Apr 2008 08:25:44 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

sukadev@us.ibm.com writes:

> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>  
> Subject: [lxc-dev] [patch -lxc 1/3] change clone\_flags type to u64  
>  
> This is a preliminary patch changing the clone\_flags type to 64bits  
> for all the routines called by do\_fork().

I must admit I was always a little sceptical of giving every tiny namespaceable kernel feature its own CLONE flag (and its own CONFIG option). What was the rationale for that again?

With your current strategy are you sure that even 64bit will be enough in the end? For me it rather looks like you'll go through those quickly too as more and more of the kernel is namespaced.

Also I think the user interface is very unfriendly. How is a non kernel hacker supposed to make sense of these myriads of flags? You'll be creating another CreateProcess123\_extra\_args\_extended() in the end I fear.

Wouldn't it be better to just partition all this into fewer more understandable larger feature groups? I think that would be much nicer from pretty much all perspectives (kernel maintenance, user interface sanity, not needing clone128/256 in the end etc.)

Some consolidation on the CONFIGs would be good too. I just cannot imagine it really makes sense to configure everything so fine grained and this is just asking for random compile breakage on randconfig.

-Andi

---

Containers mailing list

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Cedric Le Goater](#) on Thu, 10 Apr 2008 12:25:31 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Hello Andi,

Andi Kleen wrote:

> sukadev@us.ibm.com writes:

>

>> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

>> Subject: [lxc-dev] [patch -lxc 1/3] change clone\_flags type to u64

>>

>> This is a preliminary patch changing the clone\_flags type to 64bits

>> for all the routines called by do\_fork().

>

> I must admit I was always a little sceptical of giving every tiny

> namespaceable kernel feature its own CLONE flag (and it's own

> CONFIG option). What was the rationale for that again?

I guess that was a development rationale. Most of the namespaces are in use in the container projects like openvz, vserver and probably others and we needed a way to activate the code.

Not perfect I agree.

> With your current strategy are you sure that even 64bit will  
> be enough in the end? For me it rather looks like you'll  
> go through those quickly too as more and more of the kernel  
> is namespaced.

well, we're reaching the end. I hope ! devpts is in progress and mq is just waiting for a clone flag.

> Also I think the user interface is very unfriendly. How  
> is a non kernel hacker supposed to make sense of these  
> myriads of flags? You'll be creating another  
> CreateProcess123\_extra\_args\_extended()  
> in the end I fear.

well, the clone interface is a not friendly interface anyway. glibc wraps it and most users just use fork().

We will need a user library, like we have a libpthreads or a libaio, to effectively use the namespaces features. This is being worked on but

it's another topic.

- > Wouldn't it be better to just partition all this into
- > fewer more understandable larger feature groups? I think
- > that would be much nicer from pretty much all perspectives
- > (kernel maintenance, user interface sanity, not needing
- > clone128/256 in the end etc.)

Yes. this make sense. Most of the namespaces have dependencies between each other.

- > Some consolidation on the CONFIGs would be good too. I just
- > cannot imagine it really makes sense to configure everything
- > so fine grained and this is just asking for random compile
- > breakage on randconfig.

yes. definitely agree.

but we still need a way to extend the clone flags because none are left.  
would you say that the clone64 is the right way to go or should we rather go in the direction hpa proposed :

<http://lkml.org/lkml/2008/4/9/318> :

- > If you're going to make it a 64-bit pass it in as a 64-bit number,
- > instead of breaking it into two numbers. Better yet, IMO, would
- > be to pass a pointer to a structure like:
- >
- > struct shared {
- > unsigned long nwords;
- > unsigned long flags[];
- > };
- >
- > ... which can be expanded indefinitely.

if we could agree on some new interface, we could then make sure we are not abusing it.

Thanks,

C.

---

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

---

---



Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [Cedric Le Goater](#) on Thu, 10 Apr 2008 12:33:10 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

H. Peter Anvin wrote:

> sukadev@us.ibm.com wrote:

>>>

>>> If you're going to make it a 64-bit pass it in as a 64-bit number,

>>> instead of breaking it into two numbers.

>>

>> Maybe I am missing your point. The glibc interface could take a 64bit

>> parameter, but don't we need to pass 32-bit values into the system

>> call on 32 bit systems ?

>

> Not as such, no. The ABI handles that. To make the ABI clean on some

> architectures, it's good to consider a 64-bit value only in positions

> where they map to an even:odd register pair once slotted in.

OK. I didn't know that. I took sys\_llseek() as an example of an interface to follow when coded clone64().

Thanks,

C.

---

Containers mailing list

[Containers@lists.linux-foundation.org](mailto:Containers@lists.linux-foundation.org)

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Andi Kleen](#) on Thu, 10 Apr 2008 12:50:27 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

> I guess that was a development rationale.

But what rationale? It just doesn't make much sense to me.

> Most of the namespaces are in

> use in the container projects like openvz, vserver and probably others

> and we needed a way to activate the code.

You could just have added it to feature groups over time.

>

> Not perfect I agree.

>

> > With your current strategy are you sure that even 64bit will

> > be enough in the end? For me it rather looks like you'll  
> > go through those quickly too as more and more of the kernel  
> > is namespaced.  
>  
> well, we're reaching the end. I hope ! devpts is in progress and  
> mq is just waiting for a clone flag.

Are you sure?

>  
> > Also I think the user interface is very unfriendly. How  
> > is a non kernel hacker supposed to make sense of these  
> > myriads of flags? You'll be creating another  
> > CreateProcess123\_extra\_args\_extended()  
> > in the end I fear.  
>  
> well, the clone interface is a not friendly interface anyway. glibc wraps  
> it

But only for the stack setup which is just a minor detail.

The basic clone() flags interface used to be pretty sane and usable  
before it could overloaded with so many tiny features.

I especially worry on how user land should keep track of changing kernel  
here. If you add new feature flag for lots of kernel features it is  
reasonable to expect that in the future there will be often new features.

Does this mean user land needs to be updated all the time? Will this  
end up like another udev?

> We will need a user library, like we have a libpthread or a libaio, to

That doesn't make sense. The basic kernel syscalls should be usable,  
not require some magic library that would likely need intimate  
knowledge of specific kernel versions to do any good.

> but we still need a way to extend the clone flags because none are left.

Can we just take out some again that were added in the .25 cycle and  
readd them once there is a properly thought out interface? That would  
leave at least one.

-Andi

---

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

---

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Kirill Korotaev](#) on Thu, 10 Apr 2008 13:11:53 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

The was no real rationale except for some people seeing "clone" functionality as the match and the fact that FS\_NAMESCAPE was done so made them believe it is a good way to go.

And I warned about flags limitation at the beginning.

Both OpenVZ/vserver suggested to use a special syscall for handling this.

Maybe it is a good point to switch to it now finally and stop worrying about all this?

Andi Kleen wrote:

>> I guess that was a development rationale.

>

> But what rationale? It just doesn't make much sense to me.

>

>> Most of the namespaces are in

>> use in the container projects like openvz, vserver and probably others

>> and we needed a way to activate the code.

>

> You could just have added it to feature groups over time.

>

>> Not perfect I agree.

>>

>>> With your current strategy are you sure that even 64bit will

>>> be enough in the end? For me it rather looks like you'll

>>> go through those quickly too as more and more of the kernel

>>> is namespaced.

>> well, we're reaching the end. I hope ! devpts is in progress and

>> mq is just waiting for a clone flag.

>

> Are you sure?

>

>>

>>> Also I think the user interface is very unfriendly. How

>>> is a non kernel hacker supposed to make sense of these

>>> myriads of flags? You'll be creating another

>>> CreateProcess123\_extra\_args\_extended()

>>> in the end I fear.

>> well, the clone interface is a not friendly interface anyway. glibc wraps

>> it

>

> But only for the stack setup which is just a minor detail.

>

> The basic clone() flags interface used to be pretty sane and usable

> before it could overloaded with so many tiny features.

>

> I especially worry on how user land should keep track of changing kernel

> here. If you add new feature flag for lots of kernel features it is

> reasonable to expect that in the future there will be often new features.  
>  
> Does this mean user land needs to be updated all the time? Will this  
> end up like another udev?  
>  
>> We will need a user library, like we have a libpthread or a libaio, to  
>  
> That doesn't make sense. The basic kernel syscalls should be usable,  
> not require some magic library that would likely need intimate  
> knowledge of specific kernel versions to do any good.  
>  
>> but we still need a way to extend the clone flags because none are left.  
>  
> Can we just take out some again that were added in the .25 cycle and  
> readd them once there is a properly thought out interface? That would  
> leave at least one.  
>  
> -Andi  
>  
> \_\_\_\_\_  
> Containers mailing list  
> Containers@lists.linux-foundation.org  
> https://lists.linux-foundation.org/mailman/listinfo/containers  
>

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

---

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Cedric Le Goater](#) on Thu, 10 Apr 2008 13:18:20 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Andi Kleen wrote:  
>> I guess that was a development rationale.  
>  
> But what rationale? It just doesn't make much sense to me.

Let's add Eric in Cc:

>> Most of the namespaces are in  
>> use in the container projects like openvz, vserver and probably others  
>> and we needed a way to activate the code.  
>  
> You could just have added it to feature groups over time.

Yes if the feature group had existed, that would have been a good option.

Don't take me wrong. I agree with this group direction. Most namespaces can't be safely decoupled from each other with a clone flag.

>> Not perfect I agree.

>>

>>> With your current strategy are you sure that even 64bit will  
>>> be enough in the end? For me it rather looks like you'll  
>>> go through those quickly too as more and more of the kernel  
>>> is namespaced.

>> well, we're reaching the end. I hope ! devpts is in progress and  
>> mq is just waiting for a clone flag.

>

> Are you sure?

I'm never sure ! :) That's what we have in plan for the moment.

>>> Also I think the user interface is very unfriendly. How  
>>> is a non kernel hacker supposed to make sense of these  
>>> myriads of flags? You'll be creating another  
>>> CreateProcess123\_extra\_args\_extended()  
>>> in the end I fear.

>> well, the clone interface is a not friendly interface anyway. glibc wraps  
>> it

>

> But only for the stack setup which is just a minor detail.

>

> The basic clone() flags interface used to be pretty sane and usable  
> before it could overloaded with so many tiny features.

>

> I especially worry on how user land should keep track of changing kernel  
> here. If you add new feature flag for lots of kernel features it is  
> reasonable to expect that in the future there will be often new features.

>

> Does this mean user land needs to be updated all the time? Will this  
> end up like another udev?

>

>> We will need a user library, like we have a libpthread or a libaio, to

>

> That doesn't make sense. The basic kernel syscalls should be usable,  
> not require some magic library that would likely need intimate  
> knowledge of specific kernel versions to do any good.

No magic there. but running a container will require some userland code to be set up properly.

>> but we still need a way to extend the clone flags because none are left.

>  
> Can we just take out some again that were added in the .25 cycle and  
> readd them once there is a properly thought out interface? That would  
> leave at least one.

well, CLONE\_STOPPED is being recycle in 2.6.26. so we could use that one  
to group namespaces.

and CLONE\_NEWPID would probably be a good candidate to group namespaces.

That would be fine for me but it would still leave clone with one to zero  
flags left.

Thanks,

C.

---

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

---

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Cedric Le Goater](#) on Thu, 10 Apr 2008 13:23:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Kirill Korotaev wrote:

> The was no real rationale except for some people seeing "clone" functionality  
> as the match and the fact that FS\_NAMESCAPE was done so made them believe it  
> is a good way to go.  
> And I warned about flags limitation at the beginning.

yes and now, we've hit the clone wall but that's a good thing.

> Both OpenVZ/vserver suggested to use a special syscall for handling this.

most projects do it that way.

> Maybe it is a good point to switch to it now finally and stop worrying about all  
> this?

what would be the interface ?

C.

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

---

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [hpa](#) on Thu, 10 Apr 2008 16:00:05 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Cedric Le Goater wrote:

>  
> OK. I didn't know that. I took sys\_llseek() as an example of an interface  
> to follow when coded clone64().  
>

llseek() was the first system call that took a doublewidth argument.  
It's not the one you want to mimic.

-hpa

---

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

---

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [serue](#) on Thu, 10 Apr 2008 17:14:53 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Quoting Andi Kleen (andi@firstfloor.org):

> > I guess that was a development rationale.  
>  
> But what rationale? It just doesn't make much sense to me.  
>  
> > Most of the namespaces are in  
> > use in the container projects like openvz, vserver and probably others  
> > and we needed a way to activate the code.  
>  
> You could just have added it to feature groups over time.  
>  
> >  
> > Not perfect I agree.  
> >  
> > > With your current strategy are you sure that even 64bit will  
> > > be enough in the end? For me it rather looks like you'll  
> > > go through those quickly too as more and more of the kernel  
> > > is namespaced.

> >  
> > well, we're reaching the end. I hope ! devpts is in progress and  
> > mq is just waiting for a clone flag.  
>  
> Are you sure?

Well for one thing we can take a somewhat different approach to new clone flags. I.e. we could extend CLONE\_NEWIPC to do mq instead of introducing a new clone flag. The name doesn't have 'sysv' in it, and globbing all ipc resources together makes some amount of sense. Similarly has hpa+eric pointed out earlier, suka could use CLONE\_NEWDEV for ptys. If we have net, pid, ipc, devices, that's a pretty reasonable split imo. Perhaps we tie user to devices and get rid of CLONE\_NEWUSER which I suspect noone is using atm (since only Dave has run into the CONFIG\_USER\_SCHED problem). Or not. We could roll uts into net, and give CLONE\_NEWUTS a deprecation period.

-serge

---

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

---

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [Sukadev Bhattiprolu](#) on Thu, 10 Apr 2008 18:26:16 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Paul Menage [menage@google.com] wrote:  
| On Wed, Apr 9, 2008 at 7:38 PM, <sukadev@us.ibm.com> wrote:  
| >  
| > But as Jon Corbet pointed out in the the thread above, it looked like  
| > adding a new system call has been the "traditional" way of solving this  
| > in Linux so far and there has been no consensus on a newer approach.  
| >  
| I thought that the consensus was that adding a new system call was  
| better than trying to force extensibility on to the existing  
| non-extensible system call.

There were couple of objections to extensible system calls like  
sys\_indirect() and to Pavel's approach.

|  
| But if we are adding a new system call, why not make the new one  
| extensible to reduce the need for yet another new call in the future?

hypothetically, can we make a variant of clone() extensible to the point



of requiring a copy\_from\_user() ?

|  
| Paul

---

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

---

---

Subject: Re: [PATCH 0/3] clone64() and unshare64() system calls  
Posted by [hpa](#) on Thu, 10 Apr 2008 18:31:41 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

sukadev@us.ibm.com wrote:

> |  
> | I thought that the consensus was that adding a new system call was  
> | better than trying to force extensibility on to the existing  
> | non-extensible system call.  
>  
> There were couple of objections to extensible system calls like  
> sys\_indirect() and to Pavel's approach.  
>

This is a very different thing, though. sys\_indirect is pretty much a mechanism for having a sideband channel -- a second ABI -- into each and every system call, making it extremely hard to analyze what the full set of impact of a specific system call is. Worse, as it was being proposed to have been used, it would have set state variables inside the kernel in a very opaque manner.

> | But if we are adding a new system call, why not make the new one  
> | extensible to reduce the need for yet another new call in the future?  
>  
> hypothetically, can we make a variant of clone() extensible to the point  
> of requiring a copy\_from\_user() ?

The only issue is whether or not it's acceptable from a performance standpoint. clone() is reasonably expensive, though.

-hpa

---

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

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Daniel Hokka Zakrisso](#) on Thu, 10 Apr 2008 22:13:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Serge E. Hallyn wrote:

> Quoting Andi Kleen (andi@firstfloor.org):

>> > I guess that was a development rationale.

>>

>> But what rationale? It just doesn't make much sense to me.

>>

>> > Most of the namespaces are in

>> > use in the container projects like openvz, vserver and probably others

>> > and we needed a way to activate the code.

>>

>> You could just have added it to feature groups over time.

>>

>> >

>> > Not perfect I agree.

>> >

>> > > With your current strategy are you sure that even 64bit will

>> > > be enough in the end? For me it rather looks like you'll

>> > > go through those quickly too as more and more of the kernel

>> > > is namespaced.

>> >

>> > well, we're reaching the end. I hope ! devpts is in progress and

>> > mq is just waiting for a clone flag.

>>

>> Are you sure?

>

> Well for one thing we can take a somewhat different approach to new

> clone flags. I.e. we could extend CLONE\_NEWIPC to do mq instead of

> introducing a new clone flag. The name doesn't have 'sysv' in it,

> and globbing all ipc resources together makes some amount of sense.

> Similarly has hpa+eric pointed out earlier, suka could use

> CLONE\_NEWDEV for ptys. If we have net, pid, ipc, devices, that's a

> pretty reasonable split imo. Perhaps we tie user to devices and get

> rid of CLONE\_NEWUSER which I suspect noone is using atm (since only

> Dave has run into the CONFIG\_USER\_SCHED problem). Or not. We could

> roll uts into net, and give CLONE\_NEWUTS a deprecation period.

Please don't. Then we'd need to re-add it in Linux-VServer to support  
guests where network namespaces aren't used...

> -serge

--

Daniel Hokka Zakrisson

---

Containers mailing list

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [serue](#) on Thu, 10 Apr 2008 22:49:47 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Quoting Daniel Hokka Zakrisson (daniel@hozac.com):

> Serge E. Hallyn wrote:

> > Quoting Andi Kleen (andi@firstfloor.org):

> >> > I guess that was a development rationale.

> >>

> >> But what rationale? It just doesn't make much sense to me.

> >>

> >> > Most of the namespaces are in

> >> > use in the container projects like openvz, vserver and probably others

> >> > and we needed a way to activate the code.

> >>

> >> You could just have added it to feature groups over time.

> >>

> >> >

> >> > Not perfect I agree.

> >> >

> >> > > With your current strategy are you sure that even 64bit will

> >> > > be enough in the end? For me it rather looks like you'll

> >> > > go through those quickly too as more and more of the kernel

> >> > > is namespaced.

> >> >

> >> > well, we're reaching the end. I hope ! devpts is in progress and

> >> > mq is just waiting for a clone flag.

> >>

> >> Are you sure?

> >

> > Well for one thing we can take a somewhat different approach to new

> > clone flags. I.e. we could extend CLONE\_NEWIPC to do mq instead of

> > introducing a new clone flag. The name doesn't have 'sysv' in it,

> > and globbing all ipc resources together makes some amount of sense.

> > Similarly has hpa+eric pointed out earlier, suka could use

> > CLONE\_NEWDEV for ptys. If we have net, pid, ipc, devices, that's a

> > pretty reasonable split imo. Perhaps we tie user to devices and get

> > rid of CLONE\_NEWUSER which I suspect noone is using atm (since only

> > Dave has run into the CONFIG\_USER\_SCHED problem). Or not. We could

> > roll uts into net, and give CLONE\_NEWUTS a deprecation period.

>

> Please don't. Then we'd need to re-add it in Linux-VServer to support

> guests where network namespaces aren't used...

So these are networked vservers with a different hostname? Just curious, what would be a typical use for these?

Anyway then I guess we won't :) Do you have other suggestions for ns clone flags which ought to be combined? Do the rest of what I listed make sense to you? (If not, then I guess I'll step out of the way and let you and Andi fight it out :)

thanks,  
-serge

---

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

---

---

Subject: Re: [PATCH 1/3] change clone\_flags type to u64  
Posted by [Daniel Hokka Zakrisso](#) on Fri, 11 Apr 2008 08:45:22 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Serge E. Hallyn wrote:

> Quoting Daniel Hokka Zakrisson (daniel@hozac.com):

>> Serge E. Hallyn wrote:

>> > Quoting Andi Kleen (andi@firstfloor.org):

>> >> > I guess that was a development rationale.

>> >>

>> >> But what rationale? It just doesn't make much sense to me.

>> >>

>> >> > Most of the namespaces are in

>> >> > use in the container projects like openvz, vserver and probably

>> others

>> >> > and we needed a way to activate the code.

>> >>

>> >> You could just have added it to feature groups over time.

>> >>

>> >> >

>> >> > Not perfect I agree.

>> >> >

>> >> > > With your current strategy are you sure that even 64bit will

>> >> > > be enough in the end? For me it rather looks like you'll

>> >> > > go through those quickly too as more and more of the kernel

>> >> > > is namespaced.

>> >> >

>> >> > well, we're reaching the end. I hope ! devpts is in progress and

>> >> > mq is just waiting for a clone flag.

>> >>

>> >> Are you sure?

>> >

>> > Well for one thing we can take a somewhat different approach to new  
>> > clone flags. I.e. we could extend CLONE\_NEWIPC to do mq instead of  
>> > introducing a new clone flag. The name doesn't have 'sysv' in it,  
>> > and globbing all ipc resources together makes some amount of sense.  
>> > Similarly has hpa+eric pointed out earlier, suka could use  
>> > CLONE\_NEWDEV for ptys. If we have net, pid, ipc, devices, that's a  
>> > pretty reasonable split imo. Perhaps we tie user to devices and get  
>> > rid of CLONE\_NEWUSER which I suspect noone is using atm (since only  
>> > Dave has run into the CONFIG\_USER\_SCHED problem). Or not. We could  
>> > roll uts into net, and give CLONE\_NEWUTS a deprecation period.  
>>  
>> Please don't. Then we'd need to re-add it in Linux-VServer to support  
>> guests where network namespaces aren't used...  
>  
> So these are networked vservers with a different hostname? Just  
> curious, what would be a typical use for these?

Layer 3 isolation will continue to be the default for Linux-VServer.

> Anyway then I guess we won't :) Do you have other suggestions for  
> ns clone flags which ought to be combined? Do the rest of what I  
> listed make sense to you? (If not, then I guess I'll step out of the  
> way and let you and Andi fight it out :)

I think putting mq under CLONE\_NEWIPC makes sense, as well as using  
CLONE\_NEWDEV for the ptys. If CLONE\_NEWUSER is to be combined with  
anything, I think it makes more sense to combine it with CLONE\_NEWPID than  
CLONE\_NEWDEV.

> thanks,  
> -serge  
>

--

Daniel Hokka Zakrisson

---

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

---