

# [patch 03/12] syslets: generic kernel bits

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg10240.html>

- From: Ingo Molnar <mingo@xxxxxxx>
- Date: Wed, 28 Feb 2007 22:41:44 +0100

From: Ingo Molnar <mingo@xxxxxxx>

add the kernel generic bits – these are present even if !CONFIG\_ASYNC\_SUPPORT.

Signed-off-by: Ingo Molnar <mingo@xxxxxxx>  
 Signed-off-by: Arjan van de Ven <arjan@xxxxxxxxxxxxxxxxxx>

```

---
fs/exec.c | 4 ++++
include/linux/sched.h | 23 ++++++-----
kernel/capability.c | 3 +++
kernel/exit.c | 7 ++++++
kernel/fork.c | 5 +++++
kernel/sched.c | 9 ++++++
kernel/sys.c | 36 ++++++
7 files changed, 86 insertions(+), 1 deletion(-)

```

Index: linux/fs/exec.c

```

=====
--- linux.orig/fs/exec.c
+++ linux/fs/exec.c
@@ -1444,6 +1444,10 @@ static int coredump_wait(int exit_code)
tsk->vfork_done = NULL;
complete(vfork_done);
}
+ /*
+ * Make sure we exit our async context before waiting:
+ */
+ async_exit(tsk);

```

```

if (core_waiters)
wait_for_completion(&startup_done);
Index: linux/include/linux/sched.h

```

```

=====
--- linux.orig/include/linux/sched.h
+++ linux/include/linux/sched.h
@@ -83,12 +83,12 @@ struct sched_param {
#include <linux/timer.h>
#include <linux/hrtimer.h>
#include <linux/task_io_accounting.h>

```

```

+#include <linux/async.h>

#include <asm/processor.h>

struct exec_domain;
struct futex_pi_state;
-
/*
 * List of flags we want to share for kernel threads,
 * if only because they are not used by them anyway.
@@ -997,6 +997,12 @@ struct task_struct {
/* journalling filesystem info */
void *journal_info;

+/* async syscall support: */
+ struct async_thread *at, *async_ready;
+ struct async_head *ah;
+ struct async_thread __at;
+ struct async_head __ah;
+
/* VM state */
struct reclaim_state *reclaim_state;

@@ -1055,6 +1061,21 @@ struct task_struct {
#endif
};

+/*
+ * Is an async syscall being executed currently?
+ */
+#ifdef CONFIG_ASYNC_SUPPORT
+static inline int async_syscall(struct task_struct *t)
+{
+ return t->async_ready != NULL;
+}
+#else /* !CONFIG_ASYNC_SUPPORT */
+static inline int async_syscall(struct task_struct *t)
+{
+ return 0;
+}
+#endif /* !CONFIG_ASYNC_SUPPORT */
+
static inline pid_t process_group(struct task_struct *tsk)
{
return tsk->signal->pgrp;
Index: linux/kernel/capability.c
=====
--- linux.orig/kernel/capability.c
+++ linux/kernel/capability.c
@@ -178,6 +178,9 @@ asmlinkage long sys_capset(cap_user_head
int ret;

```

```
pid_t pid;

+ if (async_syscall(current))
+ return -ENOSYS;
+
if (get_user(version, &header->version))
return -EFAULT;
```

Index: linux/kernel/exit.c

```
=====
--- linux.orig/kernel/exit.c
+++ linux/kernel/exit.c
@@ -26,6 +26,7 @@
#include <linux/ptrace.h>
#include <linux/profile.h>
#include <linux/mount.h>
+#include <linux/async.h>
#include <linux/proc_fs.h>
#include <linux/mempolicy.h>
#include <linux/taskstats_kern.h>
@@ -890,6 +891,12 @@ fastcall NORET_TYPE void do_exit(long co
schedule());
}

+ /*
+ * Note: async threads have to exit their context before the MM
+ * exit (due to the coredumping wait):
+ */
+ async_exit(tsk);
+
tsk->flags |= PF_EXITING;
```

```
if (unlikely(in_atomic()))
```

Index: linux/kernel/fork.c

```
=====
--- linux.orig/kernel/fork.c
+++ linux/kernel/fork.c
@@ -22,6 +22,7 @@
#include <linux/personality.h>
#include <linux/mempolicy.h>
#include <linux/sem.h>
+#include <linux/async.h>
#include <linux/file.h>
#include <linux/key.h>
#include <linux/binfmts.h>
@@ -1056,6 +1057,7 @@ static struct task_struct *copy_process(

p->lock_depth = -1; /* -1 = no lock */
do_posix_clock_monotonic_gettime(&p->start_time);
+ async_init(p);
p->security = NULL;
```

[patch 03/12] syslets: generic kernel bits

```
p->io_context = NULL;
p->io_wait = NULL;
@@ -1623,6 +1625,9 @@ asmlinkage long sys_unshare(unsigned lon
struct uts_namespace *uts, *new_uts = NULL;
struct ipc_namespace *ipc, *new_ipc = NULL;
```

```
+ if (async_syscall(current))
+ return -ENOSYS;
+
check_unshare_flags(&unshare_flags);
```

```
/* Return -EINVAL for all unsupported flags */
```

```
Index: linux/kernel/sched.c
```

=====

```
--- linux.orig/kernel/sched.c
```

```
+++ linux/kernel/sched.c
```

```
@@ -38,6 +38,7 @@
```

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

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

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

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

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

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

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

```
@@ -3455,6 +3456,14 @@ asmlinkage void __sched schedule(void)
```

```
}
```

```
profile_hit(SCHED_PROFILING, __builtin_return_address(0));
```

```
+ prev = current;
```

```
+ if (unlikely(prev->async_ready)) {
```

```
+ if (prev->state && !(preempt_count() & PREEMPT_ACTIVE) &&
```

```
+ (!(prev->state & TASK_INTERRUPTIBLE) ||
```

```
+ !signal_pending(prev)))
```

```
+ __async_schedule(prev);
```

```
+ }
```

```
+
```

```
need_resched:
```

```
preempt_disable();
```

```
prev = current;
```

```
Index: linux/kernel/sys.c
```

=====

```
--- linux.orig/kernel/sys.c
```

```
+++ linux/kernel/sys.c
```

```
@@ -941,6 +941,9 @@ asmlinkage long sys_setregid(gid_t rgid,
```

```
int new_egid = old_egid;
```

```
int retval;
```

```
+ if (async_syscall(current))
```

```
+ return -ENOSYS;
```

```
+
```

```
retval = security_task_setgid(rgid, egid, (gid_t)-1, LSM_SETID_RE);
```

```
if (retval)
return retval;
@@ -987,6 +990,9 @@ asmlinkage long sys_setgid(gid_t gid)
int old_egid = current->egid;
int retval;

+ if (async_syscall(current))
+ return -ENOSYS;
+
retval = security_task_setgid(gid, (gid_t)-1, (gid_t)-1, LSM_SETID_ID);
if (retval)
return retval;
@@ -1057,6 +1063,9 @@ asmlinkage long sys_setreuid(uid_t ruid,
int old_ruid, old_euid, old_suid, new_ruid, new_euid;
int retval;

+ if (async_syscall(current))
+ return -ENOSYS;
+
retval = security_task_setuid(ruid, euid, (uid_t)-1, LSM_SETID_RE);
if (retval)
return retval;
@@ -1120,6 +1129,9 @@ asmlinkage long sys_setuid(uid_t uid)
int old_ruid, old_suid, new_suid;
int retval;

+ if (async_syscall(current))
+ return -ENOSYS;
+
retval = security_task_setuid(uid, (uid_t)-1, (uid_t)-1, LSM_SETID_ID);
if (retval)
return retval;
@@ -1160,6 +1172,9 @@ asmlinkage long sys_setresuid(uid_t ruid)
int old_suid = current->suid;
int retval;

+ if (async_syscall(current))
+ return -ENOSYS;
+
retval = security_task_setuid(ruid, euid, suid, LSM_SETID_RES);
if (retval)
return retval;
@@ -1214,6 +1229,9 @@ asmlinkage long sys_setresgid(gid_t rgid)
{
int retval;

+ if (async_syscall(current))
+ return -ENOSYS;
+
retval = security_task_setgid(rgid, egid, sgid, LSM_SETID_RES);
if (retval)
```

[patch 03/12] syslets: generic kernel bits

```
return retval;
@@ -1269,6 +1287,9 @@ asmlinkage long sys_setfsuid(uid_t uid)
{
int old_fsuid;

+ if (async_syscall(current))
+ return -ENOSYS;
+
old_fsuid = current->fsuid;
if (security_task_setuid(uid, (uid_t)-1, (uid_t)-1, LSM_SETID_FS))
return old_fsuid;
@@ -1298,6 +1319,9 @@ asmlinkage long sys_setfsgid(gid_t gid)
{
int old_fsgid;

+ if (async_syscall(current))
+ return -ENOSYS;
+
old_fsgid = current->fsgid;
if (security_task_setgid(gid, (gid_t)-1, (gid_t)-1, LSM_SETID_FS))
return old_fsgid;
@@ -1373,6 +1397,9 @@ asmlinkage long sys_setpgid(pid_t pid, p
struct task_struct *group_leader = current->group_leader;
int err = -EINVAL;

+ if (async_syscall(current))
+ return -ENOSYS;
+
if (!pid)
pid = group_leader->pid;
if (!pgid)
@@ -1496,6 +1523,9 @@ asmlinkage long sys_setsid(void)
pid_t session;
int err = -EPERM;

+ if (async_syscall(current))
+ return -ENOSYS;
+
write_lock_irq(&tasklist_lock);

/* Fail if I am already a session leader */
@@ -1739,6 +1769,9 @@ asmlinkage long sys_setgroups(int gidset
struct group_info *group_info;
int retval;

+ if (async_syscall(current))
+ return -ENOSYS;
+
if (!capable(CAP_SETGID))
return -EPERM;
if ((unsigned)gidsetsize > NGROUPS_MAX)
```

[patch 03/12] syslets: generic kernel bits

```
@@ -2080,6 +2113,9 @@ asmlinkage long sys_prctl(int option, un
{
long error;

+ if (async_syscall(current))
+ return -ENOSYS;
+
error = security_task_prctl(option, arg2, arg3, arg4, arg5);
if (error)
return error;
-
```

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in  
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx  
More majordomo info at <http://vger.kernel.org/majordomo-info.html>  
Please read the FAQ at <http://www.tux.org/lkml/>