

---

Subject: [PATCH 1/6] c/r: introduce ckpt\_task\_vnr(), ckpt\_pid\_vnr()

Posted by [Oren Laadan](#) on Mon, 07 Feb 2011 17:18:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

This helpoer is Useful to get the pid from the root task's (checkpoint or restart) point of view.

Signed-off-by: Oren Laadan <[orenl@cs.columbia.edu](mailto:orenl@cs.columbia.edu)>

---

include/linux/checkpoint.h | 10 ++++++++-

kernel/checkpoint/checkpoint.c | 4 ++--

kernel/checkpoint/restart.c | 6 ++----

kernel/signal.c | 2 +-

4 files changed, 14 insertions(+), 8 deletions(-)

diff --git a/include/linux/checkpoint.h b/include/linux/checkpoint.h

index 6c0ccfd..21fc23e 100644

--- a/include/linux/checkpoint.h

+++ b/include/linux/checkpoint.h

@@ -114,8 +114,16 @@ extern int checkpoint\_dump\_page(struct ckpt\_ctx \*ctx, struct page \*page);

extern int restore\_read\_page(struct ckpt\_ctx \*ctx, struct page \*page);

/\* pids \*/

-extern pid\_t ckpt\_pid\_nr(struct ckpt\_ctx \*ctx, struct pid \*pid);

extern struct pid \*\_ckpt\_find\_pgrp(struct ckpt\_ctx \*ctx, pid\_t pgid);

+static inline pid\_t ckpt\_task\_vnr(struct ckpt\_ctx \*ctx, struct task\_struct \*task)

{

+ return task\_pid\_nr\_ns(task, ctx->root\_nsproxy->pid\_ns);

}

+static inline pid\_t ckpt\_pid\_vnr(struct ckpt\_ctx \*ctx, struct pid \*pid)

{

+ return pid\_nr\_ns(pid, ctx->root\_nsproxy->pid\_ns);

}

+extern int ckpt\_lookup\_pid(struct ckpt\_ctx \*ctx, struct pid \*pid);

/\* defined in objhash.c and also used in security/security.c \*/

extern void lsm\_string\_free(struct kref \*kref);

diff --git a/kernel/checkpoint/checkpoint.c b/kernel/checkpoint/checkpoint.c

index 0f46acf..01653d7 100644

--- a/kernel/checkpoint/checkpoint.c

+++ b/kernel/checkpoint/checkpoint.c

@@ -243,10 +243,10 @@ static int may\_checkpoint\_task(struct ckpt\_ctx \*ctx, struct task\_struct \*t)

{

struct task\_struct \*root = ctx->root\_task;

struct nsproxy \*nsproxy;

- int ret = 0;

```

    struct pid_namespace *pidns;
+ int ret = 0;

- ckpt_debug("check %d\n", task_pid_nr_ns(t, ctx->root_nsproxy->pid_ns));
+ ckpt_debug("check %d\n", ckpt_task_vnr(ctx, t));

    if (t->exit_state == EXIT_DEAD) {
        _ckpt_err(ctx, -EBUSY, "%(T)Task state EXIT_DEAD\n");
diff --git a/kernel/checkpoint/restart.c b/kernel/checkpoint/restart.c
index 01da67f..66b6625 100644
--- a/kernel/checkpoint/restart.c
+++ b/kernel/checkpoint/restart.c
@@ -913,8 +913,7 @@ static int wait_task_active(struct ckpt_ctx *ctx)

static int wait_task_sync(struct ckpt_ctx *ctx)
{
- ckpt_debug("pid %d syncing\n",
- task_pid_nr_ns(current, task_active_pid_ns(ctx->root_task)));
+ ckpt_debug("pid %d syncing\n", ckpt_task_vnr(ctx, current));
    wait_event_interruptible(ctx->waitq, ckpt_test_complete(ctx));
    ckpt_debug("task sync done (errno %d)\n", ctx->errno);
    if (ckpt_test_error(ctx))
@@ -1187,10 +1186,9 @@ static struct task_struct *choose_root_task(struct ckpt_ctx *ctx, pid_t
pid)

    read_lock(&tasklist_lock);
    list_for_each_entry(task, &current->children, sibling) {
- if (task_pid_nr_ns(task, ctx->coord_pidns) == pid) {
+ if (task_pid_vnr(task) == pid) {
        get_task_struct(task);
        ctx->root_task = task;
- ctx->root_pid = pid;
        break;
    }
}
diff --git a/kernel/signal.c b/kernel/signal.c
index b1e6a31..dca40be 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -3352,7 +3352,7 @@ static int restore_signal(struct ckpt_ctx *ctx)
    * fail, so no need for explicit test
    */
    ret = do_tiocspgrp(tty, tty_pair_get_tty(tty),
- h->tty_pgrp);
+ pid_vnr(pgrp));
    if (ret < 0)
        goto out;
}

```

--

1.7.1

---

Containers mailing list

Containers@lists.linux-foundation.org

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

---