Subject: Re: [RFC][PATCH 1/5] Virtualization/containers: startup
Posted by Hubertus Franke on Fri, 03 Feb 2006 19:56:02 GMT
View Forum Message <> Reply to Message

missed the typedef on container ..
I am still on 2.6.15.2

> Couple of minor naming nitpick questions, though. Is vps/container_info
> really what we want to call it? It seems to me to be the basis for a
> real "container", without the _info part.
>
> "tsk->owner_container" That makes it sound like a pointer to the "task
> owner's container". How about "owning_container"? The "container
> owning this task". Or, maybe just "container"?
>
> Any particular reason for the "u32 id" in the vps_info struct as opposed
> to one of the more generic types? Do we want to abstract this one in
> the same way we do pid_t?
>
> The "host" in "host_container_info" doesn't mean much to me. Though, I
> guess it has some context in the UML space. Would "init_container_info"
> or "root_container_info" be more descriptive?
>
> Lastly, is this a place for krefs? I don't see a real need for a
> destructor yet, but the idea is fresh in my mind.
>
> How does the attached patch look?

here is that adapted ....

Index: linux-2.6.15/include/linux/container.h
========== =======
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.15/include/linux/container.h 2006-02-03 14:52:12.000000000
-0500
@@ -0,0 +1,42 @@
+#ifndef __LINUX_CONTAINER_H_
+#define __LINUX_CONTAINER_H_
+
+#include <asm/types.h>
+#include <asm/atomic.h>
+
+struct task_struct;
+
+struct container {
+ u32 id;
+ struct task_struct *init_task;
+ atomic_t refcnt;

```
+};
+
+extern struct container root_container;
+
+static inline struct container*
+get_container(struct container *container)
+{
+ atomic_inc(&container->refcnt);
+ return container;
+}
+
+static inline void put_container(struct container *container)
+{
+ atomic_dec(&container->refcnt);
+}
+
+#include <linux/sched.h>
+#include <asm/current.h>
+
+static inline struct container*
+set_current_container(struct container *container)
+{
+ struct container *ret;
+
+ ret = current->container;
+ current->container = container;
+ return ret;
+}
+
+#endif
Index: linux-2.6.15/include/linux/init_task.h
 ============================================================== =======
--- linux-2.6.15.orig/include/linux/init_task.h 2006-02-03
14:50:39.000000000 -0500
+++ linux-2.6.15/include/linux/init_task.h 2006-02-03 14:52:12.000000000
-0500
@@ -3,6 +3,7 @@

 #include <linux/file.h>
 #include <linux/rcupdate.h>
+#include <linux/container.h>

 #define INIT_FDTABLE \
 {       \
@@ -121,6 +122,7 @@ extern struct group_info init_groups;
  .journal_info = NULL,      \
  .cpu_timers = INIT_CPU_TIMERS(tsk.cpu_timers),  \
  .fs_excl = ATOMIC_INIT(0),    \
```

```
+ .container = &root_container,    \
 }


Index: linux-2.6.15/include/linux/sched.h
 ===================================================================
--- linux-2.6.15.orig/include/linux/sched.h 2006-02-03
14:50:39.000000000 -0500
+++ linux-2.6.15/include/linux/sched.h 2006-02-03 14:52:12.000000000
-0500
@@ -682,6 +682,7 @@ static inline void prefetch_stack(struct

 struct audit_context;  /* See audit.c */
 struct mempolicy;
+struct container;

 struct task_struct {
  volatile long state; /* -1 unrunnable, 0 runnable, >0 stopped */
@@ -830,6 +831,7 @@ struct task_struct {
  struct backing_dev_info *backing_dev_info;

  struct io_context *io_context;
+ struct container *container;

  unsigned long ptrace_message;
  siginfo_t *last_siginfo; /* For ptrace use.  */
@@ -859,6 +861,8 @@ struct task_struct {
  atomic_t fs_excl; /* holding fs exclusive resources */
 };

+#define current_container() (current->container)
+
 static inline pid_t process_group(struct task_struct *tsk)
 {
  return tsk->signal->pgrp;
Index: linux-2.6.15/init/main.c
 ===================================================================
--- linux-2.6.15.orig/init/main.c 2006-02-03 14:50:39.000000000 -0500
+++ linux-2.6.15/init/main.c 2006-02-03 14:52:12.000000000 -0500
@@ -47,8 +47,8 @@
 #include <linux/rmap.h>
 #include <linux/mempolicy.h>
 #include <linux/key.h>
+#include <linux/container.h>
 #include <net/sock.h>
-
 #include <asm/io.h>
 #include <asm/bugs.h>
```

```
 #include <asm/setup.h>
@@ -438,6 +438,13 @@ void __init parse_early_param(void)
  done = 1;
 }

+struct container root_container = {
+ .id  = 0,
+ .init_task = &init_task,
+ .refcnt  = ATOMIC_INIT(1),
+};
+
+EXPORT_SYMBOL(root_container);
 /*
  * Activate the first processor.
  */
Index: linux-2.6.15/kernel/exit.c
 ===================================================================
--- linux-2.6.15.orig/kernel/exit.c 2006-02-03 14:50:39.000000000 -0500
+++ linux-2.6.15/kernel/exit.c 2006-02-03 14:52:12.000000000 -0500
@@ -29,6 +29,7 @@
 #include <linux/syscalls.h>
 #include <linux/signal.h>
 #include <linux/cn_proc.h>
+#include <linux/container.h>

 #include <asm/uaccess.h>
 #include <asm/unistd.h>
@@ -106,6 +107,7 @@ repeat:
  spin_unlock(&p->proc_lock);
  proc_pid_flush(proc_dentry);
  release_thread(p);
+ put_container(p->container);
  put_task_struct(p);

  p = leader;
Index: linux-2.6.15/kernel/fork.c
 ===================================================================
--- linux-2.6.15.orig/kernel/fork.c 2006-02-03 14:50:39.000000000 -0500
+++ linux-2.6.15/kernel/fork.c 2006-02-03 14:52:12.000000000 -0500
@@ -43,6 +43,7 @@
 #include <linux/rmap.h>
 #include <linux/acct.h>
 #include <linux/cn_proc.h>
+#include <linux/container.h>

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -1121,6 +1122,7 @@ static task_t *copy_process(unsigned lon
```

```
  p->ioprio = current->ioprio;

  SET_LINKS(p);
+ get_container(p->container);
  if (unlikely(p->ptrace & PT_PTRACED))
   __ptrace_link(p, current->parent);
```