Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by Pavel Emelianov on Mon, 10 Sep 2007 13:16:16 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:

[snip]

> --- /dev/null
> +++ b/include/net/net_namespace.h
> @@ -0,0 +1,68 @@
> +/*
> + * Operations on the network namespace
> + */
> +#ifndef __NET_NET_NAMESPACE_H
> +#define __NET_NET_NAMESPACE_H
> +
> +#include <asm/atomic.h>
> +#include <linux/workqueue.h>
> +#include <linux/list.h>
> +
> +struct net {

Isn't this name is too generic? Why not net_namespace?

> + atomic_t count; /* To decided when the network
> +   * namespace should be freed.
> +   */
> + atomic_t use_count; /* To track references we
> +   * destroy on demand
> +   */
> + struct list_head list; /* list of network namespaces */
> + struct work_struct work; /* work struct for freeing */
> +};

[snip]

> --- /dev/null
> +++ b/net/core/net_namespace.c

[snip]

> +static int setup_net(struct net *net)
> +{
> + /* Must be called with net_mutex held */
> + struct pernet_operations *ops;
> + struct list_head *ptr;
> + int error;

```
> +
> + memset(net, 0, sizeof(struct net));
> + atomic_set(&net->count, 1);
> + atomic_set(&net->use_count, 0);
> +
> + error = 0;
> + list_for_each(ptr, &pernet_list) {
> +  ops = list_entry(ptr, struct pernet_operations, list);
> +  if (ops->init) {
> +   error = ops->init(net);
> +   if (error < 0)
> +    goto out_undo;
> +  }
> + }
> +out:
> + return error;
> +out_undo:
> + /* Walk through the list backwards calling the exit functions
> +  * for the pernet modules whose init functions did not fail.
> +  */
> + for (ptr = ptr->prev; ptr != &pernet_list; ptr = ptr->prev) {
```

Good reason for adding list_for_each_continue_reverse :)

```
> +  ops = list_entry(ptr, struct pernet_operations, list);
> +  if (ops->exit)
> +   ops->exit(net);
> + }
> + goto out;
> +}
> +
> +static int __init net_ns_init(void)
> +{
> + int err;
> +
> + printk(KERN_INFO "net_namespace: %zd bytes\n", sizeof(struct net));
> + net_cachep = kmem_cache_create("net_namespace", sizeof(struct net),
> +     SMP_CACHE_BYTES,
> +     SLAB_PANIC, NULL);
> + mutex_lock(&net_mutex);
> + err = setup_net(&init_net);
> +
> + net_lock();
> + list_add_tail(&init_net.list, &net_namespace_list);
> + net_unlock();
> +
> + mutex_unlock(&net_mutex);
> + if (err)
```

```
> +  panic("Could not setup the initial network namespace");
> +
> + return 0;
> +}
> +
> +pure_initcall(net_ns_init);
> +
> +static int register_pernet_operations(struct list_head *list,
> +          struct pernet_operations *ops)
> +{
> + struct net *net, *undo_net;
> + int error;
> +
> + error = 0;
> + list_add_tail(&ops->list, list);
> + for_each_net(net) {
> +  if (ops->init) {

Maybe it's better to do it vice-versa?
if (ops->init)
   for_each_net(net)
       ops->init(net);
...

> +   error = ops->init(net);
> +   if (error)
> +    goto out_undo;
> +  }
> + }
> +out:
> + return error;
> +
> +out_undo:
> + /* If I have an error cleanup all namespaces I initialized */
> + list_del(&ops->list);
> + for_each_net(undo_net) {
> +  if (undo_net == net)
> +   goto undone;
> +  if (ops->exit)
> +   ops->exit(undo_net);
> + }
> +undone:
> + goto out;
> +}
> +
> +static void unregister_pernet_operations(struct pernet_operations *ops)
> +{
> + struct net *net;
```

> +
> + list_del(&ops->list);
> + for_each_net(net)
> + if (ops->exit)

The same here.

> +   ops->exit(net);
> +}
> +
> +/**
> + *   register_pernet_subsys - register a network namespace subsystem
> + * @ops: pernet operations structure for the subsystem
> + *
> + * Register a subsystem which has init and exit functions
> + * that are called when network namespaces are created and
> + * destroyed respectively.
> + *
> + * When registered all network namespace init functions are
> + * called for every existing network namespace. Allowing kernel
> + * modules to have a race free view of the set of network namespaces.
> + *
> + * When a new network namespace is created all of the init
> + * methods are called in the order in which they were registered.
> + *
> + * When a network namespace is destroyed all of the exit methods
> + * are called in the reverse of the order with which they were
> + * registered.
> + */
> +int register_pernet_subsys(struct pernet_operations *ops)
> +{
> + int error;
> + mutex_lock(&net_mutex);
> + error = register_pernet_operations(first_device, ops);
> + mutex_unlock(&net_mutex);
> + return error;
> +}
> +EXPORT_SYMBOL_GPL(register_pernet_subsys);
> +
> +/**
> + *   unregister_pernet_subsys - unregister a network namespace subsystem
> + * @ops: pernet operations structure to manipulate
> + *
> + * Remove the pernet operations structure from the list to be
> + * used when network namespaces are created or destoryed. In
> + * addition run the exit method for all existing network
> + * namespaces.
> + */

```
> +void unregister_pernet_subsys(struct pernet_operations *module)
> +{
> + mutex_lock(&net_mutex);
> + unregister_pernet_operations(module);
> + mutex_unlock(&net_mutex);
> +}
> +EXPORT_SYMBOL_GPL(unregister_pernet_subsys);
> +
> +/**
> + *      register_pernet_device - register a network namespace device
> + * @ops:  pernet operations structure for the subsystem
> + *
> + * Register a device which has init and exit functions
> + * that are called when network namespaces are created and
> + * destroyed respectively.
> + *
> + * When registered all network namespace init functions are
> + * called for every existing network namespace.  Allowing kernel
> + * modules to have a race free view of the set of network namespaces.
> + *
> + * When a new network namespace is created all of the init
> + * methods are called in the order in which they were registered.
> + *
> + * When a network namespace is destroyed all of the exit methods
> + * are called in the reverse of the order with which they were
> + * registered.
> + */
> +int register_pernet_device(struct pernet_operations *ops)
> +{
> + int error;
> + mutex_lock(&net_mutex);
> + error = register_pernet_operations(&pernet_list, ops);
> + if (!error && (first_device == &pernet_list))
```

Very minor: why do you give the name "device" to some pernet_operations?

Thanks,
Pavel