Subject: [PATCH 2.6.25] netns: struct net content re-work
Posted by den on Mon, 10 Dec 2007 16:35:45 GMT
View Forum Message <> Reply to Message

Recently David Miller and Herbert Xu pointed out that struct net becomes
overbloated and un-maintainable. There are two solutions:
- provide a pointer to a network subsystem definition from struct net.
  This costs an additional dereferrence
- place sub-system definition into the structure itself. This will speedup
  run-time access at the cost of recompilation time

The second approach looks better for us. Other sub-systems will be converted
to this approach if this will be accepted :)

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
index b62e31f..f60e1ce 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -8,6 +8,8 @@
 #include <linux/workqueue.h>
 #include <linux/list.h>

+#include <net/netns/unix.h>
+
 struct proc_dir_entry;
 struct net_device;
 struct sock;
@@ -46,8 +48,7 @@ struct net {
  struct hlist_head packet_sklist;

  /* unix sockets */
- int   sysctl_unix_max_dgram_qlen;
- struct ctl_table_header *unix_ctl;
+ struct netns_unix unx;
 };

 #ifdef CONFIG_NET
diff --git a/include/net/netns/unix.h b/include/net/netns/unix.h
new file mode 100644
index 0000000..27b4e7f
--- /dev/null
+++ b/include/net/netns/unix.h
@@ -0,0 +1,13 @@
+/*
+ * Unix network namespace
+ */

```
+#ifndef __NETNS_UNIX_H__
+#define __NETNS_UNIX_H__
+
+struct ctl_table_header;
+struct netns_unix {
+ int   sysctl_unix_max_dgram_qlen;
+ struct ctl_table_header *unix_ctl;
+};
+
+#endif /* __NETNS_UNIX_H__ */
diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c
index b8a2189..06f7ec8 100644
--- a/net/unix/af_unix.c
+++ b/net/unix/af_unix.c
@@ -592,7 +592,7 @@ static struct sock * unix_create1(struct net *net, struct socket *sock)
     &af_unix_sk_receive_queue_lock_key);

  sk->sk_write_space = unix_write_space;
- sk->sk_max_ack_backlog = net->sysctl_unix_max_dgram_qlen;
+ sk->sk_max_ack_backlog = net->unx.sysctl_unix_max_dgram_qlen;
  sk->sk_destruct  = unix_sock_destructor;
  u   = unix_sk(sk);
  u->dentry = NULL;
@@ -2138,7 +2138,7 @@ static int unix_net_init(struct net *net)
 {
  int error = -ENOMEM;

- net->sysctl_unix_max_dgram_qlen = 10;
+ net->unx.sysctl_unix_max_dgram_qlen = 10;
  if (unix_sysctl_register(net))
   goto out;

diff --git a/net/unix/sysctl_net_unix.c b/net/unix/sysctl_net_unix.c
index 553ef6a..fbddfb5 100644
--- a/net/unix/sysctl_net_unix.c
+++ b/net/unix/sysctl_net_unix.c
@@ -18,7 +18,7 @@ static ctl_table unix_table[] = {
  {
   .ctl_name = NET_UNIX_MAX_DGRAM_QLEN,
   .procname = "max_dgram_qlen",
-  .data  = &init_net.sysctl_unix_max_dgram_qlen,
+  .data  = &init_net.unx.sysctl_unix_max_dgram_qlen,
   .maxlen  = sizeof(int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec
@@ -40,9 +40,9 @@ int unix_sysctl_register(struct net *net)
  if (table == NULL)
   goto err_alloc;
```

```
- table[0].data = &net->sysctl_unix_max_dgram_qlen;
- net->unix_ctl = register_net_sysctl_table(net, unix_path, table);
- if (net->unix_ctl == NULL)
+ table[0].data = &net->unx.sysctl_unix_max_dgram_qlen;
+ net->unx.unix_ctl = register_net_sysctl_table(net, unix_path, table);
+ if (net->unx.unix_ctl == NULL)
   goto err_reg;

  return 0;
@@ -57,8 +57,8 @@ void unix_sysctl_unregister(struct net *net)
 {
  struct ctl_table *table;

- table = net->unix_ctl->ctl_table_arg;
- unregister_sysctl_table(net->unix_ctl);
+ table = net->unx.unix_ctl->ctl_table_arg;
+ unregister_sysctl_table(net->unx.unix_ctl);
  kfree(table);
 }
```

Subject: Re: [PATCH 2.6.25] netns: struct net content re-work
Posted by Daniel Lezcano on Mon, 10 Dec 2007 17:32:57 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Recently David Miller and Herbert Xu pointed out that struct net becomes
> overbloated and un-maintainable. There are two solutions:
> - provide a pointer to a network subsystem definition from struct net.
>   This costs an additional dereferrence
> - place sub-system definition into the structure itself. This will speedup
>   run-time access at the cost of recompilation time
>
> The second approach looks better for us.

Yes, we do not need/want a pointer in this structure and add more
dereference in the network code.

> Other sub-systems will be converted
> to this approach if this will be accepted :)
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>
> ---
> diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
> index b62e31f..f60e1ce 100644
> --- a/include/net/net_namespace.h
> +++ b/include/net/net_namespace.h

> @@ -8,6 +8,8 @@
> #include <linux/workqueue.h>
> #include <linux/list.h>
>
> +#include <net/netns/unix.h>
> +
> struct proc_dir_entry;
> struct net_device;
> struct sock;
> @@ -46,8 +48,7 @@ struct net {
>   struct hlist_head packet_sklist;
>
>   /* unix sockets */
> - int   sysctl_unix_max_dgram_qlen;
> - struct ctl_table_header *unix_ctl;
> + struct netns_unix unx;

Can you change this from unx to unix ?

If you encapsulate the structure definitions per subsystem, you can drop
the unix prefix in the variable declaration.

Instead of having:
 netns->unix->unix_ctl
you will have:
 netns->unix->ctl

> };
>
> #ifdef CONFIG_NET
> diff --git a/include/net/netns/unix.h b/include/net/netns/unix.h
> new file mode 100644
> index 0000000..27b4e7f
> --- /dev/null
> +++ b/include/net/netns/unix.h
> @@ -0,0 +1,13 @@
> +/*
> + * Unix network namespace
> + */
> +#ifndef __NETNS_UNIX_H__
> +#define __NETNS_UNIX_H__
> +
> +struct ctl_table_header;
> +struct netns_unix {
> + int   sysctl_unix_max_dgram_qlen;
> + struct ctl_table_header *unix_ctl;
> +};
> +

> +#endif /* __NETNS_UNIX_H__ */

If I follow the logic, we will have a file per subsystem. These files
will be very small, no ?
IMHO, having the structure defined in net_namespace.h is enough.

> diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c
> index b8a2189..06f7ec8 100644
> --- a/net/unix/af_unix.c
> +++ b/net/unix/af_unix.c
> @@ -592,7 +592,7 @@ static struct sock * unix_create1(struct net *net, struct socket *sock)
> &af_unix_sk_receive_queue_lock_key);
>
> sk->sk_write_space = unix_write_space;
> - sk->sk_max_ack_backlog = net->sysctl_unix_max_dgram_qlen;
> + sk->sk_max_ack_backlog = net->unx.sysctl_unix_max_dgram_qlen;
> sk->sk_destruct = unix_sock_destructor;
> u = unix_sk(sk);
> u->dentry = NULL;
> @@ -2138,7 +2138,7 @@ static int unix_net_init(struct net *net)
> {
> int error = -ENOMEM;
>
> - net->sysctl_unix_max_dgram_qlen = 10;
> + net->unx.sysctl_unix_max_dgram_qlen = 10;
> if (unix_sysctl_register(net))
> goto out;
>
> diff --git a/net/unix/sysctl_net_unix.c b/net/unix/sysctl_net_unix.c
> index 553ef6a..fbddfb5 100644
> --- a/net/unix/sysctl_net_unix.c
> +++ b/net/unix/sysctl_net_unix.c
> @@ -18,7 +18,7 @@ static ctl_table unix_table[] = {
> {
> .ctl_name = NET_UNIX_MAX_DGRAM_QLEN,
> .procname = "max_dgram_qlen",
> - .data = &init_net.sysctl_unix_max_dgram_qlen,
> + .data = &init_net.unx.sysctl_unix_max_dgram_qlen,
> .maxlen = sizeof(int),
> .mode = 0644,
> .proc_handler = &proc_dointvec
> @@ -40,9 +40,9 @@ int unix_sysctl_register(struct net *net)
> if (table == NULL)
> goto err_alloc;
>
> - table[0].data = &net->sysctl_unix_max_dgram_qlen;
> - net->unix_ctl = register_net_sysctl_table(net, unix_path, table);
> - if (net->unix_ctl == NULL)

> + table[0].data = &net->unx.sysctl_unix_max_dgram_qlen;
> + net->unx.unix_ctl = register_net_sysctl_table(net, unix_path, table);
> + if (net->unx.unix_ctl == NULL)
>   goto err_reg;
>
>   return 0;
> @@ -57,8 +57,8 @@ void unix_sysctl_unregister(struct net *net)
> {
>   struct ctl_table *table;
>
> - table = net->unix_ctl->ctl_table_arg;
> - unregister_sysctl_table(net->unix_ctl);
> + table = net->unx.unix_ctl->ctl_table_arg;
> + unregister_sysctl_table(net->unx.unix_ctl);
>   kfree(table);
> }
>
> --
> To unsubscribe from this list: send the line "unsubscribe netdev" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
>

Subject: Re: [PATCH 2.6.25] netns: struct net content re-work
Posted by dev on Mon, 10 Dec 2007 17:47:39 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Denis V. Lunev wrote:
>
>>Recently David Miller and Herbert Xu pointed out that struct net becomes
>>overbloated and un-maintainable. There are two solutions:
>>- provide a pointer to a network subsystem definition from struct net.
>> This costs an additional dereferrence
>>- place sub-system definition into the structure itself. This will speedup
>> run-time access at the cost of recompilation time
>>
>>The second approach looks better for us.
>
>
> Yes, we do not need/want a pointer in this structure and add more
> dereference in the network code.
>
>
>>Other sub-systems will be converted
>>to this approach if this will be accepted :)
>>

>>Signed-off-by: Denis V. Lunev <den@openvz.org>
>>---
>>diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
>>index b62e31f..f60e1ce 100644
>>--- a/include/net/net_namespace.h
>>+++ b/include/net/net_namespace.h
>>@@ -8,6 +8,8 @@
>> #include <linux/workqueue.h>
>> #include <linux/list.h>
>>
>>+#include <net/netns/unix.h>
>>+
>> struct proc_dir_entry;
>> struct net_device;
>> struct sock;
>>@@ -46,8 +48,7 @@ struct net {
>> struct hlist_head packet_sklist;
>>
>> /* unix sockets */
>>- int   sysctl_unix_max_dgram_qlen;
>>- struct ctl_table_header *unix_ctl;
>>+ struct netns_unix unx;
>>
>>
> Can you change this from unx to unix ?

no, it won't compile. Guess why :)

> If you encapsulate the structure definitions per subsystem, you can drop
> the unix prefix in the variable declaration.
>
> Instead of having:
> netns->unix->unix_ctl
> you will have:
> netns->unix->ctl

agree.

Kirill

Subject: Re: [PATCH 2.6.25] netns: struct net content re-work
Posted by ebiederm on Tue, 11 Dec 2007 03:52:56 GMT

The idea of separate structures make sense, and seems needed and useful.

"Denis V. Lunev" <den@openvz.org> writes:

> diff --git a/include/net/netns/unix.h b/include/net/netns/unix.h
> new file mode 100644
> index 0000000..27b4e7f
> --- /dev/null
> +++ b/include/net/netns/unix.h
^^^^^^

Given that we are making this per protocol adding a separate directory
to hold them seems to be the wrong grouping. Ideally we want everything
for the protocol all together in the same location so it is easy
to find. Possibly with a user/kernel split.

So perhaps unix_net.h

> @@ -0,0 +1,13 @@
> +/*
> + * Unix network namespace
> + */
> +#ifndef __NETNS_UNIX_H__
> +#define __NETNS_UNIX_H__
> +
> +struct ctl_table_header;
> +struct netns_unix {
> + int sysctl_unix_max_dgram_qlen;
> + struct ctl_table_header *unix_ctl;
> +};

How about struct unix_net? I think that tracks a little better
with how we have done struct in_device, ip6_dev and their friends.

Eric

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

Subject: Re: [PATCH 2.6.25] netns: struct net content re-work
Posted by ebiederm on Tue, 11 Dec 2007 04:04:07 GMT
View Forum Message <> Reply to Message

Kirill Korotaev <dev@sw.ru> writes:

> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>
>>>Recently David Miller and Herbert Xu pointed out that struct net becomes
>>>overbloated and un-maintainable. There are two solutions:
>>>- provide a pointer to a network subsystem definition from struct net.
>>> This costs an additional dereferrence
>>>- place sub-system definition into the structure itself. This will speedup
>>> run-time access at the cost of recompilation time
>>>
>>>The second approach looks better for us.
>>
>>
>> Yes, we do not need/want a pointer in this structure and add more
>> dereference in the network code.

If it does go that way we just carefully pass around a properly
typed structure in that subsystem to reduce the cost. Still
it would be nice not to need to add the extra pointer.

>>>index b62e31f..f60e1ce 100644
>>>--- a/include/net/net_namespace.h
>>>+++ b/include/net/net_namespace.h
>>>@@ -8,6 +8,8 @@
>>> #include <linux/workqueue.h>
>>> #include <linux/list.h>
>>>
>>>+#include <net/netns/unix.h>
>>>+
>>> struct proc_dir_entry;
>>> struct net_device;
>>> struct sock;
>>>@@ -46,8 +48,7 @@ struct net {
>>> struct hlist_head packet_sklist;
>>>
>>> /* unix sockets */
>>>- int   sysctl_unix_max_dgram_qlen;
>>>- struct ctl_table_header *unix_ctl;
>>>+ struct netns_unix unx;
>>>
>>>
>> Can you change this from unx to unix ?
>
> no, it won't compile. Guess why :)

Hmm. It looks like it is a #define somewhere gcc?
Eric

Subject: Re: [PATCH 2.6.25] netns: struct net content re-work
Posted by davem on Tue, 11 Dec 2007 04:12:06 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Mon, 10 Dec 2007 21:04:07 -0700

> Kirill Korotaev <dev@sw.ru> writes:
>
> > Daniel Lezcano wrote:
> >> Denis V. Lunev wrote:
> >> Can you change this from unx to unix ?
> >
> > no, it won't compile. Guess why :)
>
> Hmm. It looks like it is a #define somewhere gcc?

It is a platform CPP pre-define for UNIX platforms.

Subject: Re: [PATCH 2.6.25] netns: struct net content re-work
Posted by den on Tue, 11 Dec 2007 07:33:45 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> The idea of separate structures make sense, and seems needed and useful.
>
> "Denis V. Lunev" <den@openvz.org> writes:
>
>> diff --git a/include/net/netns/unix.h b/include/net/netns/unix.h
>> new file mode 100644
>> index 0000000..27b4e7f
>> --- /dev/null
>> +++ b/include/net/netns/unix.h
>   ^^^^^^
> Given that we are making this per protocol adding a separate directory
> to hold them seems to be the wrong grouping. Ideally we want everything
> for the protocol all together in the same location so it is easy

> to find. Possibly with a user/kernel split.
>
> So perhaps unix_net.h
The idea was simple:
- I can name 5 files right now
- I want them to be shown to gather by ls
- so, there are 2 ways, namely:
  # include/net/netns/unix.h
  # include/net/netns-unix.h

Regards,
 Den

>
>> @@ -0,0 +1,13 @@
>> +/*
>> + * Unix network namespace
>> + */
>> +#ifndef __NETNS_UNIX_H__
>> +#define __NETNS_UNIX_H__
>> +
>> +struct ctl_table_header;
>> +struct netns_unix {
>> + int   sysctl_unix_max_dgram_qlen;
>> + struct ctl_table_header *unix_ctl;
>> +};
>
> How about struct unix_net? I think that tracks a little better
> with how we have done struct in_device, ip6_dev and their friends.
>
> Eric
>

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

---

Subject: Re: [PATCH 2.6.25] netns: struct net content re-work
Posted by davem on Tue, 11 Dec 2007 10:27:24 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@sw.ru>
Date: Tue, 11 Dec 2007 10:33:45 +0300

> Eric W. Biederman wrote:
> > The idea of separate structures make sense, and seems needed and useful.

> >
> > "Denis V. Lunev" <den@openvz.org> writes:
> >
> >> diff --git a/include/net/netns/unix.h b/include/net/netns/unix.h
> >> new file mode 100644
> >> index 0000000..27b4e7f
> >> --- /dev/null
> >> +++ b/include/net/netns/unix.h
> >   ^^^^^^
> > Given that we are making this per protocol adding a separate directory
> > to hold them seems to be the wrong grouping. Ideally we want everything
> > for the protocol all together in the same location so it is easy
> > to find. Possibly with a user/kernel split.
> >
> > So perhaps unix_net.h
> The idea was simple:
> - I can name 5 files right now
> - I want them to be shown to gather by ls
> - so, there are 2 ways, namely:
>   # include/net/netns/unix.h
>   # include/net/netns-unix.h

I have no real objection to the new directory.

_____

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