Subject: [PATCH RFC 21/31] net: Implement the guts of the network namespace infrastructure
Posted by ebiederm on Thu, 25 Jan 2007 19:00:23 GMT
View Forum Message <> Reply to Message

From: Eric W. Biederman <ebiederm@xmission.com> - unquoted

Support is added for the .data.pernet section where all of
the variables who have a single instance in each network
namespace will live. Every architectures linker script
is modified so is should work.

Summarizing the functions:
net_ns_init creates a slab and allocates the template and
the initial network namespace.

pernet_modcopy keeps the network namespaces in sync with
the loaded modules. Initializing new data variables as
they are added.

The network namespace destruction because the last reference
can come from interrupt context queues itself for later with
schedule_work. Then we alert everyone the network namespace
is disappearing. If a buggy user is still holding a reference
to the network namespace we print a nasty message and leak
the network namespace.

The wrest are just light-weight wrapper functions to make things
more convinient.

A little should probably be said about net_head the variable
at the start of my network namespace structure. It is the only
variable with a location decided by the C code instead of the linker
and I string them together in a linked list so I can iterate.

Probably more interesting is that it looks like it is saner not
to directly use a pointer to my network namespace but instead to
use an offset. All of the references to data in my network namespace
are coming from per_net(...) which takes the address of the variable
in the .data.pernet section and then adds my magic offset. If I
used a pointer I would have to subract an additional value and
export an extra symbol. Not good for performance or maintenance :)

The expected usage of network namespace variables is to replace
sequences like: &loopback_dev with &per_net(loopback_dev, net)
where net is some network namespace reference. In my preliminary
tests the only a single additional addition is inserted so it
appears to be an efficient idiom. Hopefully it is also easy to

comprehend and use.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

```
 arch/alpha/kernel/vmlinux.lds.S            |    2 +
 arch/arm/kernel/vmlinux.lds.S              |    3 +
 arch/arm26/kernel/vmlinux-arm26-xip.lds.in |    3 +
 arch/arm26/kernel/vmlinux-arm26.lds.in     |    3 +
 arch/avr32/kernel/vmlinux.lds.c            |    3 +
 arch/cris/arch-v10/vmlinux.lds.S           |    2 +
 arch/cris/arch-v32/vmlinux.lds.S           |    2 +
 arch/frv/kernel/vmlinux.lds.S              |    2 +
 arch/h8300/kernel/vmlinux.lds.S            |    3 +
 arch/i386/kernel/vmlinux.lds.S             |    3 +
 arch/ia64/kernel/vmlinux.lds.S             |    2 +
 arch/m32r/kernel/vmlinux.lds.S             |    3 +
 arch/m68k/kernel/vmlinux-std.lds           |    3 +
 arch/m68k/kernel/vmlinux-sun3.lds          |    3 +
 arch/m68knommu/kernel/vmlinux.lds.S        |    3 +
 arch/mips/kernel/vmlinux.lds.S             |    3 +
 arch/parisc/kernel/vmlinux.lds.S           |    3 +
 arch/powerpc/kernel/vmlinux.lds.S          |    2 +
 arch/ppc/kernel/vmlinux.lds.S              |    2 +
 arch/s390/kernel/vmlinux.lds.S             |    3 +
 arch/sh/kernel/vmlinux.lds.S               |    3 +
 arch/sh64/kernel/vmlinux.lds.S             |    3 +
 arch/sparc/kernel/vmlinux.lds.S            |    3 +
 arch/sparc64/kernel/vmlinux.lds.S          |    3 +
 arch/v850/kernel/vmlinux.lds.S             |    6 +-
 arch/x86_64/kernel/vmlinux.lds.S           |    3 +
 arch/xtensa/kernel/vmlinux.lds.S           |    2 +
 include/asm-generic/vmlinux.lds.h          |    8 +
 include/asm-um/common.lds.S                |    4 +-
 include/linux/module.h                     |    3 +
 include/linux/net_namespace_type.h         |   63 ++++++++-
 include/net/net_namespace.h                |   49 ++++++-
 kernel/module.c                            |  211 ++++++++++++++++++++++++-
 net/core/net_namespace.c                   |  232 ++++++++++++++++++++++++++++
 34 files changed, 631 insertions(+), 15 deletions(-)

diff --git a/arch/alpha/kernel/vmlinux.lds.S b/arch/alpha/kernel/vmlinux.lds.S
index 76bf071..ad20077 100644
--- a/arch/alpha/kernel/vmlinux.lds.S
+++ b/arch/alpha/kernel/vmlinux.lds.S
@@ -72,6 +72,8 @@ SECTIONS
   .data.percpu : { *(.data.percpu) }
   __per_cpu_end = .;
```

```
+  DATA_PER_NET
+
   . = ALIGN(2*8192);
   __init_end = .;
   /* Freed after init ends here */
diff --git a/arch/arm/kernel/vmlinux.lds.S b/arch/arm/kernel/vmlinux.lds.S
index a8fa75e..5b003f9 100644
--- a/arch/arm/kernel/vmlinux.lds.S
+++ b/arch/arm/kernel/vmlinux.lds.S
@@ -61,6 +61,9 @@ SECTIONS
   __per_cpu_start = .;
    *(.data.percpu)
   __per_cpu_end = .;
+
+  DATA_PER_NET
+
  #ifndef CONFIG_XIP_KERNEL
   __init_begin = _stext;
   *(.init.data)
diff --git a/arch/arm26/kernel/vmlinux-arm26-xip.lds.in
b/arch/arm26/kernel/vmlinux-arm26-xip.lds.in
index ca61ec8..69d5772 100644
--- a/arch/arm26/kernel/vmlinux-arm26-xip.lds.in
+++ b/arch/arm26/kernel/vmlinux-arm26-xip.lds.in
@@ -50,6 +50,9 @@ SECTIONS
   __initramfs_start = .;
    usr/built-in.o(.init.ramfs)
   __initramfs_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(32768);
   __init_end = .;
  }
diff --git a/arch/arm26/kernel/vmlinux-arm26.lds.in b/arch/arm26/kernel/vmlinux-arm26.lds.in
index d1d3418..473a5b4 100644
--- a/arch/arm26/kernel/vmlinux-arm26.lds.in
+++ b/arch/arm26/kernel/vmlinux-arm26.lds.in
@@ -51,6 +51,9 @@ SECTIONS
   __initramfs_start = .;
    usr/built-in.o(.init.ramfs)
   __initramfs_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(32768);
   __init_end = .;
  }
```

```
diff --git a/arch/avr32/kernel/vmlinux.lds.c b/arch/avr32/kernel/vmlinux.lds.c
index 5c4424e..dee3715 100644
--- a/arch/avr32/kernel/vmlinux.lds.c
+++ b/arch/avr32/kernel/vmlinux.lds.c
@@ -50,6 +50,9 @@ SECTIONS
   __initramfs_start = .;
    *(.init.ramfs)
   __initramfs_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(4096);
   __init_end = .;
  }
diff --git a/arch/cris/arch-v10/vmlinux.lds.S b/arch/cris/arch-v10/vmlinux.lds.S
index 689729a..f1c890c 100644
--- a/arch/cris/arch-v10/vmlinux.lds.S
+++ b/arch/cris/arch-v10/vmlinux.lds.S
@@ -83,6 +83,8 @@ SECTIONS
  }
  SECURITY_INIT

+ DATA_PER_NET
+
  .init.ramfs : {
   __initramfs_start = .;
   *(.init.ramfs)
diff --git a/arch/cris/arch-v32/vmlinux.lds.S b/arch/cris/arch-v32/vmlinux.lds.S
index 472d4b3..eb08771 100644
--- a/arch/cris/arch-v32/vmlinux.lds.S
+++ b/arch/cris/arch-v32/vmlinux.lds.S
@@ -95,6 +95,8 @@ SECTIONS
  .data.percpu  : { *(.data.percpu) }
  __per_cpu_end = .;

+ DATA_PER_NET
+
  .init.ramfs : {
   __initramfs_start = .;
   *(.init.ramfs)
diff --git a/arch/frv/kernel/vmlinux.lds.S b/arch/frv/kernel/vmlinux.lds.S
index 9c1fb12..f383c83 100644
--- a/arch/frv/kernel/vmlinux.lds.S
+++ b/arch/frv/kernel/vmlinux.lds.S
@@ -61,6 +61,8 @@ SECTIONS
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;
```

```
+  DATA_PER_NET
+
   . = ALIGN(4096);
   __initramfs_start = .;
   .init.ramfs : { *(.init.ramfs) }
diff --git a/arch/h8300/kernel/vmlinux.lds.S b/arch/h8300/kernel/vmlinux.lds.S
index f05288b..5d5fda5 100644
--- a/arch/h8300/kernel/vmlinux.lds.S
+++ b/arch/h8300/kernel/vmlinux.lds.S
@@ -130,6 +130,9 @@ SECTIONS
  ___initramfs_start = .;
     *(.init.ramfs)
    ___initramfs_end = .;
+
+ DATA_PER_NET
+
  . = ALIGN(0x4) ;
  ___init_end = .;
  __edata = . ;
diff --git a/arch/i386/kernel/vmlinux.lds.S b/arch/i386/kernel/vmlinux.lds.S
index a53c8b1..1aae8b4 100644
--- a/arch/i386/kernel/vmlinux.lds.S
+++ b/arch/i386/kernel/vmlinux.lds.S
@@ -193,6 +193,9 @@ SECTIONS
  *(.data.percpu)
  __per_cpu_end = .;
   }
+
+  DATA_PER_NET
+
   . = ALIGN(4096);
   /* freed after init ends here */

diff --git a/arch/ia64/kernel/vmlinux.lds.S b/arch/ia64/kernel/vmlinux.lds.S
index d6083a0..28dd9eb 100644
--- a/arch/ia64/kernel/vmlinux.lds.S
+++ b/arch/ia64/kernel/vmlinux.lds.S
@@ -118,6 +118,8 @@ SECTIONS
    __initramfs_end = .;
  }

+  DATA_PER_NET
+
    . = ALIGN(16);
   .init.setup : AT(ADDR(.init.setup) - LOAD_OFFSET)
         {
diff --git a/arch/m32r/kernel/vmlinux.lds.S b/arch/m32r/kernel/vmlinux.lds.S
index 358b9ce..3e8c624 100644
```

```
--- a/arch/m32r/kernel/vmlinux.lds.S
+++ b/arch/m32r/kernel/vmlinux.lds.S
@@ -107,6 +107,9 @@ SECTIONS
   __per_cpu_start = .;
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(4096);
   __init_end = .;
   /* freed after init ends here */
diff --git a/arch/m68k/kernel/vmlinux-std.lds b/arch/m68k/kernel/vmlinux-std.lds
index d279445..d60cb7e 100644
--- a/arch/m68k/kernel/vmlinux-std.lds
+++ b/arch/m68k/kernel/vmlinux-std.lds
@@ -65,6 +65,9 @@ SECTIONS
   __initramfs_start = .;
   .init.ramfs : { *(.init.ramfs) }
   __initramfs_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(8192);
   __init_end = .;

diff --git a/arch/m68k/kernel/vmlinux-sun3.lds b/arch/m68k/kernel/vmlinux-sun3.lds
index 8c7eccb..101ec12 100644
--- a/arch/m68k/kernel/vmlinux-sun3.lds
+++ b/arch/m68k/kernel/vmlinux-sun3.lds
@@ -59,6 +59,9 @@ __init_begin = .;
  __initramfs_start = .;
  .init.ramfs : { *(.init.ramfs) }
  __initramfs_end = .;
+
+ DATA_PER_NET
+
  . = ALIGN(8192);
  __init_end = .;
  .data.init.task : { *(.data.init_task) }
diff --git a/arch/m68knommu/kernel/vmlinux.lds.S b/arch/m68knommu/kernel/vmlinux.lds.S
index 2b2a10d..e713614 100644
--- a/arch/m68knommu/kernel/vmlinux.lds.S
+++ b/arch/m68knommu/kernel/vmlinux.lds.S
@@ -153,6 +153,9 @@ SECTIONS {
   __initramfs_start = .;
   *(.init.ramfs)
   __initramfs_end = .;
```

```
+
+  DATA_PER_NET
+
   . = ALIGN(4096);
   __init_end = .;
  } > INIT
diff --git a/arch/mips/kernel/vmlinux.lds.S b/arch/mips/kernel/vmlinux.lds.S
index cecff24..a5cfeef 100644
--- a/arch/mips/kernel/vmlinux.lds.S
+++ b/arch/mips/kernel/vmlinux.lds.S
@@ -121,6 +121,9 @@ SECTIONS
   __per_cpu_start = .;
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(_PAGE_SIZE);
   __init_end = .;
   /* freed after init ends here */
diff --git a/arch/parisc/kernel/vmlinux.lds.S b/arch/parisc/kernel/vmlinux.lds.S
index 7b943b4..2cf241b 100644
--- a/arch/parisc/kernel/vmlinux.lds.S
+++ b/arch/parisc/kernel/vmlinux.lds.S
@@ -181,6 +181,9 @@ SECTIONS
   __per_cpu_start = .;
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(ASM_PAGE_SIZE);
   __init_end = .;
   /* freed after init ends here */
diff --git a/arch/powerpc/kernel/vmlinux.lds.S b/arch/powerpc/kernel/vmlinux.lds.S
index 04b8e71..bdd4f05 100644
--- a/arch/powerpc/kernel/vmlinux.lds.S
+++ b/arch/powerpc/kernel/vmlinux.lds.S
@@ -150,6 +150,8 @@ SECTIONS
   __per_cpu_end = .;
  }

+ DATA_PER_NET
+
  . = ALIGN(8);
  .machine.desc : {
   __machine_desc_start = . ;
diff --git a/arch/ppc/kernel/vmlinux.lds.S b/arch/ppc/kernel/vmlinux.lds.S
```

```
index 6192126..59c5e6c 100644
--- a/arch/ppc/kernel/vmlinux.lds.S
+++ b/arch/ppc/kernel/vmlinux.lds.S
@@ -135,6 +135,8 @@ SECTIONS
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;

+  DATA_PER_NET
+
   . = ALIGN(4096);
   __initramfs_start = .;
   .init.ramfs : { *(.init.ramfs) }
diff --git a/arch/s390/kernel/vmlinux.lds.S b/arch/s390/kernel/vmlinux.lds.S
index fe0f2e9..bcdd353 100644
--- a/arch/s390/kernel/vmlinux.lds.S
+++ b/arch/s390/kernel/vmlinux.lds.S
@@ -99,6 +99,9 @@ SECTIONS
   __per_cpu_start = .;
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(4096);
   __init_end = .;
   /* freed after init ends here */
diff --git a/arch/sh/kernel/vmlinux.lds.S b/arch/sh/kernel/vmlinux.lds.S
index f34bdcc..0a4249d 100644
--- a/arch/sh/kernel/vmlinux.lds.S
+++ b/arch/sh/kernel/vmlinux.lds.S
@@ -86,6 +86,9 @@ SECTIONS
   __initramfs_start = .;
   .init.ramfs : { *(.init.ramfs) }
   __initramfs_end = .;
+
+  DATA_PER_NET
+
   __machvec_start = .;
   .init.machvec : { *(.init.machvec) }
   __machvec_end = .;
diff --git a/arch/sh64/kernel/vmlinux.lds.S b/arch/sh64/kernel/vmlinux.lds.S
index 95c4d75..0c1a30e 100644
--- a/arch/sh64/kernel/vmlinux.lds.S
+++ b/arch/sh64/kernel/vmlinux.lds.S
@@ -118,6 +118,9 @@ SECTIONS
   __initramfs_start = .;
   .init.ramfs : C_PHYS(.init.ramfs) { *(.init.ramfs) }
   __initramfs_end = .;
```

```
+
+  DATA_PER_NET
+
   . = ALIGN(PAGE_SIZE);
   __init_end = .;

diff --git a/arch/sparc/kernel/vmlinux.lds.S b/arch/sparc/kernel/vmlinux.lds.S
index b73e6b9..c1ff7de 100644
--- a/arch/sparc/kernel/vmlinux.lds.S
+++ b/arch/sparc/kernel/vmlinux.lds.S
@@ -65,6 +65,9 @@ SECTIONS
   __per_cpu_start = .;
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(4096);
   __init_end = .;
   . = ALIGN(32);
diff --git a/arch/sparc64/kernel/vmlinux.lds.S b/arch/sparc64/kernel/vmlinux.lds.S
index 4a6063f..24e6b7f 100644
--- a/arch/sparc64/kernel/vmlinux.lds.S
+++ b/arch/sparc64/kernel/vmlinux.lds.S
@@ -89,6 +89,9 @@ SECTIONS
   __per_cpu_start = .;
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(8192);
   __init_end = .;
   __bss_start = .;
diff --git a/arch/v850/kernel/vmlinux.lds.S b/arch/v850/kernel/vmlinux.lds.S
index 3a5fd07..b87a4cb 100644
--- a/arch/v850/kernel/vmlinux.lds.S
+++ b/arch/v850/kernel/vmlinux.lds.S
@@ -163,7 +163,8 @@
    *(.text.init) /* 2.4 convention */       \
    *(.data.init)          \
   INITCALL_CONTENTS          \
-  INITRAMFS_CONTENTS
+  INITRAMFS_CONTENTS          \
+  DATA_PER_NET

 /* The contents of `init' section for a ROM-resident kernel which
    should go into RAM.  */
```

```
@@ -183,7 +184,8 @@
    _einittext = .;          \
    *(.text.init) /* 2.4 convention */       \
   INITCALL_CONTENTS          \
-  INITRAMFS_CONTENTS
+  INITRAMFS_CONTENTS          \
+  DATA_PER_NET

 /* A root filesystem image, for kernels with an embedded root filesystem.  */
 #define ROOT_FS_CONTENTS          \
diff --git a/arch/x86_64/kernel/vmlinux.lds.S b/arch/x86_64/kernel/vmlinux.lds.S
index 1e54ddf..38061b2 100644
--- a/arch/x86_64/kernel/vmlinux.lds.S
+++ b/arch/x86_64/kernel/vmlinux.lds.S
@@ -200,6 +200,9 @@ SECTIONS
   __per_cpu_start = .;
   .data.percpu  : AT(ADDR(.data.percpu) - LOAD_OFFSET) { *(.data.percpu) }
   __per_cpu_end = .;
+
+  DATA_PER_NET
+
   . = ALIGN(4096);
   __init_end = .;

diff --git a/arch/xtensa/kernel/vmlinux.lds.S b/arch/xtensa/kernel/vmlinux.lds.S
index a36c104..e77ed43 100644
--- a/arch/xtensa/kernel/vmlinux.lds.S
+++ b/arch/xtensa/kernel/vmlinux.lds.S
@@ -203,6 +203,8 @@ SECTIONS
   .data.percpu  : { *(.data.percpu) }
   __per_cpu_end = .;

+  DATA_PER_NET
+
   . = ALIGN(4096);
   __initramfs_start =.;
   .init.ramfs : { *(.init.ramfs) }
diff --git a/include/asm-generic/vmlinux.lds.h b/include/asm-generic/vmlinux.lds.h
index 9fcc8d9..298ed43 100644
--- a/include/asm-generic/vmlinux.lds.h
+++ b/include/asm-generic/vmlinux.lds.h
@@ -229,3 +229,11 @@
    *(.initcall7.init)      \
    *(.initcall7s.init)

+#define DATA_PER_NET       \
+ .data.pernet : AT(ADDR(.data.pernet) - LOAD_OFFSET) {  \
+  VMLINUX_SYMBOL(__per_net_start) = .;   \
```

```
+  *(.data.pernet.head)     \
+  *(.data.pernet)      \
+  VMLINUX_SYMBOL(__per_net_end) = .;   \
+ }
+
diff --git a/include/asm-um/common.lds.S b/include/asm-um/common.lds.S
index f045451..1208960 100644
--- a/include/asm-um/common.lds.S
+++ b/include/asm-um/common.lds.S
@@ -39,7 +39,9 @@
   __per_cpu_start = . ;
   .data.percpu : { *(.data.percpu) }
   __per_cpu_end = . ;
-
+
+  DATA_PER_NET
+
   __initcall_start = .;
   .initcall.init : {
  INITCALLS
diff --git a/include/linux/module.h b/include/linux/module.h
index 10f771a..755f1b5 100644
--- a/include/linux/module.h
+++ b/include/linux/module.h
@@ -353,6 +353,9 @@ struct module
  /* Per-cpu data. */
  void *percpu;

+ /* Per-net data. */
+ void *pernet;
+
  /* The command line arguments (may be mangled).  People like
     keeping pointers to this stuff */
  char *args;
diff --git a/include/linux/net_namespace_type.h b/include/linux/net_namespace_type.h
index 8173f59..5075199 100644
--- a/include/linux/net_namespace_type.h
+++ b/include/linux/net_namespace_type.h
@@ -7,14 +7,70 @@

 #define __pernetname(name) per_net__##name

+#ifdef CONFIG_NET_NS
+
+typedef struct {
+ unsigned long offset;
+} net_t;
+
```

```
+#define __data_pernet __attribute__((__section__(".data.pernet")))
+
+static inline unsigned long __per_net_offset(net_t net) { return net.offset; }
+
+/* Like per_net but returns a pseudo variable address that must be offset
+ * __per_net_offset() bytes before it will point to a real variable.
+ * Useful for static initializers.
+ */
+#define __per_net_base(name)   __pernetname(name)
+
+/* Get the network namespace reference from a per_net variable address */
+#define net_of(ptr, name)      \
+({         \
+ net_t net = { .offset = 0 };     \
+ char *__ptr = (void *)(ptr);     \
+ if (__ptr)       \
+  net.offset = __ptr - ((char *)&__per_net_base(name)); \
+ net;        \
+})
+
+/* Look up a per network namespace variable */
+#define per_net(var, net) (*(     \
+ RELOC_HIDE(&__per_net_base(var), __per_net_offset(net))))
+
+/* A more efficient form if gcc doesn't overoptimize it */
+#ifndef per_net
+#define per_net(var, net) (*(     \
+ (typeof(__pernetname(var)) *)    \
+ (((char *)&__per_net_base(var)) + __per_net_offset(net))))
+#endif
+
+
+/* Are the two network namespaces the same */
+static inline int net_eq(net_t a, net_t b) { return a.offset == b.offset; }
+
+/* Get an unsigned value appropriate for hashing the network namespace */
+static inline unsigned int net_hval(net_t net) { return net.offset; }
+
+/* Convert to and from to and from void pointers */
+static inline void *net_to_voidp(net_t net) { return (void *)net.offset; }
+static inline net_t net_from_voidp(void *ptr)
+{
+ net_t r;
+ r.offset = (unsigned long)ptr;
+ return r;
+}
+
+static inline int null_net(net_t net) { return net.offset == 0; }
```

```
+
+#else /* CONFIG_NET_NS */
+
 typedef struct {} net_t;

 #define __data_pernet

-/* Look up a per network namespace variable */
 static inline unsigned long __per_net_offset(net_t net) { return 0; }

-/* Like per_net but returns a pseudo variable address that must be moved
+/* Like per_net but returns a pseudo variable address that must be offset
  * __per_net_offset() bytes before it will point to a real variable.
  * Useful for static initializers.
  */
@@ -38,6 +94,9 @@ static inline net_t net_from_voidp(void *ptr) { net_t net; return net; }

 static inline int null_net(net_t net) { return 0; }

+#endif /* CONFIG_NET_NS */
+
+
 #define DEFINE_PER_NET(type, name)    \
  __data_pernet __typeof__(type) __pernetname(name)

diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
index b64568f..a2042ac 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -24,7 +24,8 @@ struct net_namespace_head {
      * should go
      */
  atomic_t use_count; /* For references we destroy on demand */
- struct list_head list;
+ net_t next;
+ net_t prev;
  struct work_struct work;
 };

@@ -34,6 +35,50 @@ static inline net_t init_net(void)
  return init_nsproxy.net_ns;
 }

+#ifdef CONFIG_NET_NS
+
+DECLARE_PER_NET(struct net_namespace_head, net_head);
+
+extern void pernet_modcopy(void *, const void *, unsigned long);
```

```
+extern int copy_net(int, struct task_struct *);
+extern void __put_net(net_t net);
+
+static inline net_t get_net(net_t net)
+{
+ atomic_inc(&per_net(net_head, net).count);
+ return net;
+}
+
+static inline void put_net(net_t net)
+{
+ if (atomic_dec_and_test(&per_net(net_head, net).count))
+  __put_net(net);
+}
+
+static inline net_t hold_net(net_t net)
+{
+ atomic_inc(&per_net(net_head, net).use_count);
+ return net;
+}
+
+static inline void release_net(net_t net)
+{
+ atomic_dec(&per_net(net_head, net).use_count);
+}
+
+/* Created by linker magic */
+extern char __per_net_start[], __per_net_end[];
+
+extern void net_lock(void);
+extern void net_unlock(void);
+
+#define for_each_net(VAR)    \
+ for ( (VAR) = init_net(); !null_net((VAR)); \
+  (VAR) = per_net(net_head, (VAR)).next)
+
+
+#else /* CONFIG_NET_NS */
+
 static inline net_t get_net(net_t net) { return net; }
 static inline void put_net(net_t net) {}
 static inline net_t hold_net(net_t net) { return net; }
@@ -50,6 +95,8 @@ static inline void net_unlock(void) {}

 #define for_each_net(VAR) if (1)

+#endif /* CONFIG_NET_NS */
+
```

```
 extern net_t net_template;

 #define NET_CREATE 0x0001 /* A network namespace has been created */
diff --git a/kernel/module.c b/kernel/module.c
index d0f2260..6f45090 100644
--- a/kernel/module.c
+++ b/kernel/module.c
@@ -44,6 +44,7 @@
 #include <asm/semaphore.h>
 #include <asm/cacheflush.h>
 #include <linux/license.h>
+#include <net/net_namespace.h>

 #if 0
 #define DEBUGP printk
@@ -304,7 +305,7 @@ static unsigned int pcpu_num_used, pcpu_num_allocated;
 /* Size of each block.  -ve means used. */
 static int *pcpu_size;

-static int split_block(unsigned int i, unsigned short size)
+static int pcpu_split_block(unsigned int i, unsigned short size)
 {
  /* Reallocation required? */
  if (pcpu_num_used + 1 > pcpu_num_allocated) {
@@ -329,7 +330,7 @@ static int split_block(unsigned int i, unsigned short size)
  return 1;
 }

-static inline unsigned int block_size(int val)
+static inline unsigned int pcpu_block_size(int val)
 {
  if (val < 0)
   return -val;
@@ -353,7 +354,7 @@ static void *percpu_modalloc(unsigned long size, unsigned long align,
  }

  ptr = __per_cpu_start;
- for (i = 0; i < pcpu_num_used; ptr += block_size(pcpu_size[i]), i++) {
+ for (i = 0; i < pcpu_num_used; ptr += pcpu_block_size(pcpu_size[i]), i++) {
   /* Extra for alignment requirement. */
   extra = ALIGN((unsigned long)ptr, align) - (unsigned long)ptr;
   BUG_ON(i == 0 && extra != 0);
@@ -371,7 +372,7 @@ static void *percpu_modalloc(unsigned long size, unsigned long align,

   /* Split block if warranted */
   if (pcpu_size[i] - size > sizeof(unsigned long))
-   if (!split_block(i, size))
+   if (!pcpu_split_block(i, size))
```

```
     return NULL;

   /* Mark allocated */
@@ -387,10 +388,10 @@ static void *percpu_modalloc(unsigned long size, unsigned long align,
 static void percpu_modfree(void *freeme)
 {
  unsigned int i;
- void *ptr = __per_cpu_start + block_size(pcpu_size[0]);
+ void *ptr = __per_cpu_start + pcpu_block_size(pcpu_size[0]);

  /* First entry is core kernel percpu data. */
- for (i = 1; i < pcpu_num_used; ptr += block_size(pcpu_size[i]), i++) {
+ for (i = 1; i < pcpu_num_used; ptr += pcpu_block_size(pcpu_size[i]), i++) {
   if (ptr == freeme) {
    pcpu_size[i] = -pcpu_size[i];
    goto free;
@@ -465,6 +466,169 @@ static inline void percpu_modcopy(void *pcpudst, const void *src,
 }
 #endif /* CONFIG_SMP */

+#ifdef CONFIG_NET_NS
+/* Number of blocks used and allocated. */
+static unsigned int pnet_num_used, pnet_num_allocated;
+/* Size of each block.  -ve means used. */
+static int *pnet_size;
+
+static int pnet_split_block(unsigned int i, unsigned short size)
+{
+ /* Reallocation required? */
+ if (pnet_num_used + 1 > pnet_num_allocated) {
+  int *new = kmalloc(sizeof(new[0]) * pnet_num_allocated*2,
+       GFP_KERNEL);
+  if (!new)
+   return 0;
+
+  memcpy(new, pnet_size, sizeof(new[0])*pnet_num_allocated);
+  pnet_num_allocated *= 2;
+  kfree(pnet_size);
+  pnet_size = new;
+ }
+
+ /* Insert a new subblock */
+ memmove(&pnet_size[i+1], &pnet_size[i],
+  sizeof(pnet_size[0]) * (pnet_num_used - i));
+ pnet_num_used++;
+
+ pnet_size[i+1] -= size;
+ pnet_size[i] = size;
```

```
+ return 1;
+}
+
+static inline unsigned int pnet_block_size(int val)
+{
+ if (val < 0)
+  return -val;
+ return val;
+}
+
+static void *pernet_modalloc(unsigned long size, unsigned long align,
+        const char *name)
+{
+ unsigned long extra;
+ unsigned int i;
+ void *ptr;
+
+ if (align > SMP_CACHE_BYTES) {
+  printk(KERN_WARNING "%s: per-net alignment %li > %i\n",
+         name, align, SMP_CACHE_BYTES);
+  align = SMP_CACHE_BYTES;
+ }
+
+ ptr = __per_net_start;
+ for (i = 0; i < pnet_num_used; ptr += pnet_block_size(pnet_size[i]), i++) {
+  /* Extra for alignment requirement. */
+  extra = ALIGN((unsigned long)ptr, align) - (unsigned long)ptr;
+  BUG_ON(i == 0 && extra != 0);
+
+  if (pnet_size[i] < 0 || pnet_size[i] < extra + size)
+   continue;
+
+  /* Transfer extra to previous block. */
+  if (pnet_size[i-1] < 0)
+   pnet_size[i-1] -= extra;
+  else
+   pnet_size[i-1] += extra;
+  pnet_size[i] -= extra;
+  ptr += extra;
+
+  /* Split block if warranted */
+  if (pnet_size[i] - size > sizeof(unsigned long))
+   if (!pnet_split_block(i, size))
+    return NULL;
+
+  /* Mark allocated */
+  pnet_size[i] = -pnet_size[i];
+  return ptr;
```

```
+ }
+
+ printk(KERN_WARNING "Could not allocate %lu bytes pernet data\n",
+        size);
+ return NULL;
+}
+
+static void pernet_modfree(void *freeme)
+{
+ unsigned int i;
+ void *ptr = __per_net_start + pnet_block_size(pnet_size[0]);
+
+ /* First entry is core kernel pernet data. */
+ for (i = 1; i < pnet_num_used; ptr += pnet_block_size(pnet_size[i]), i++) {
+  if (ptr == freeme) {
+   pnet_size[i] = -pnet_size[i];
+   goto free;
+  }
+ }
+ BUG();
+
+ free:
+ /* Merge with previous? */
+ if (pnet_size[i-1] >= 0) {
+  pnet_size[i-1] += pnet_size[i];
+  pnet_num_used--;
+  memmove(&pnet_size[i], &pnet_size[i+1],
+   (pnet_num_used - i) * sizeof(pnet_size[0]));
+  i--;
+ }
+ /* Merge with next? */
+ if (i+1 < pnet_num_used && pnet_size[i+1] >= 0) {
+  pnet_size[i] += pnet_size[i+1];
+  pnet_num_used--;
+  memmove(&pnet_size[i+1], &pnet_size[i+2],
+   (pnet_num_used - (i+1)) * sizeof(pnet_size[0]));
+ }
+}
+
+static unsigned int find_pnetsec(Elf_Ehdr *hdr,
+     Elf_Shdr *sechdrs,
+     const char *secstrings)
+{
+ return find_sec(hdr, sechdrs, secstrings, ".data.pernet");
+}
+
+static int pernet_modinit(void)
+{
```

```
+ pnet_num_used = 2;
+ pnet_num_allocated = 2;
+ pnet_size = kmalloc(sizeof(pnet_size[0]) *pnet_num_allocated,
+       GFP_KERNEL);
+ /* Static in-kernel pernet data (used). */
+ pnet_size[0] = -ALIGN(__per_net_end-__per_net_start, SMP_CACHE_BYTES);
+ /* Free room. */
+ pnet_size[1] = PER_NET_MODULE_RESERVE;
+ if (pnet_size[1] <= 0) {
+  printk(KERN_ERR "No per-net room for modules.\n");
+  pnet_num_used = 1 ;
+ }
+ return 0;
+}
+__initcall(pernet_modinit);
+#else /* ... !CONFIG_NET_NS */
+static inline void *pernet_modalloc(unsigned long size, unsigned long align,
+        const char *name)
+{
+ return NULL;
+}
+static inline void pernet_modfree(void *pnetptr)
+{
+ BUG();
+}
+static inline unsigned int find_pnetsec(Elf_Ehdr *hdr,
+     Elf_Shdr *sechdrs,
+     const char *secstrings)
+{
+ return 0;
+}
+static inline void pernet_modcopy(void *pnetdst, const void *src,
+      unsigned long size)
+{
+ /* pnetsec should be 0, and size of that section should be 0. */
+ BUG_ON(size != 0);
+}
+#endif /* CONFIG_NET_NS */
+
 #define MODINFO_ATTR(field) \
 static void setup_modinfo_##field(struct module *mod, const char *s)  \
 {                                                                     \
@@ -1198,6 +1362,8 @@ static void free_module(struct module *mod)
  /* This may be NULL, but that's OK */
  module_free(mod, mod->module_init);
  kfree(mod->args);
+ if (mod->pernet)
+  pernet_modfree(mod->pernet);
```

```
  if (mod->percpu)
   percpu_modfree(mod->percpu);

@@ -1263,6 +1429,7 @@ static int simplify_symbols(Elf_Shdr *sechdrs,
        const char *strtab,
        unsigned int versindex,
        unsigned int pcpuindex,
+       unsigned int pnetindex,
        struct module *mod)
 {
  Elf_Sym *sym = (void *)sechdrs[symindex].sh_addr;
@@ -1308,6 +1475,9 @@ static int simplify_symbols(Elf_Shdr *sechdrs,
    /* Divert to percpu allocation if a percpu var. */
    if (sym[i].st_shndx == pcpuindex)
     secbase = (unsigned long)mod->percpu;
+   /* Divert to pernet allocation if a pernet var. */
+   else if (sym[i].st_shndx == pnetindex)
+    secbase = (unsigned long)mod->pernet;
    else
     secbase = sechdrs[sym[i].st_shndx].sh_addr;
    sym[i].st_value += secbase;
@@ -1554,6 +1724,7 @@ static struct module *load_module(void __user *umod,
  unsigned int gplcrcindex;
  unsigned int versindex;
  unsigned int pcpuindex;
+ unsigned int pnetindex;
  unsigned int gplfutureindex;
  unsigned int gplfuturecrcindex;
  unsigned int unwindex = 0;
@@ -1563,7 +1734,7 @@ static struct module *load_module(void __user *umod,
  unsigned int unusedgplcrcindex;
  struct module *mod;
  long err = 0;
- void *percpu = NULL, *ptr = NULL; /* Stops spurious gcc warning */
+ void *percpu = NULL, *pernet = NULL, *ptr = NULL; /* Stops spurious gcc warning */
  struct exception_table_entry *extable;
  mm_segment_t old_fs;

@@ -1654,6 +1825,7 @@ static struct module *load_module(void __user *umod,
  versindex = find_sec(hdr, sechdrs, secstrings, "__versions");
  infoindex = find_sec(hdr, sechdrs, secstrings, ".modinfo");
  pcpuindex = find_pcpusec(hdr, sechdrs, secstrings);
+ pnetindex = find_pnetsec(hdr, sechdrs, secstrings);
 #ifdef ARCH_UNWIND_SECTION_NAME
  unwindex = find_sec(hdr, sechdrs, secstrings, ARCH_UNWIND_SECTION_NAME);
 #endif
@@ -1719,6 +1891,20 @@ static struct module *load_module(void __user *umod,
   mod->percpu = percpu;
```

```
  }

+ if (pnetindex) {
+  /* We have a special allocation for this section */
+  pernet = pernet_modalloc(sechdrs[pnetindex].sh_size,
+      sechdrs[pnetindex].sh_addralign,
+      mod->name);
+
+  if (!pernet) {
+   err = -ENOMEM;
+   goto free_percpu;
+  }
+  sechdrs[pnetindex].sh_flags &= ~(unsigned long)SHF_ALLOC;
+  mod->pernet = pernet;
+ }
+
  /* Determine total sizes, and put offsets in sh_entsize.  For now
     this is done generically; there doesn't appear to be any
     special cases for the architectures. */
@@ -1728,7 +1914,7 @@ static struct module *load_module(void __user *umod,
  ptr = module_alloc(mod->core_size);
  if (!ptr) {
   err = -ENOMEM;
-  goto free_percpu;
+  goto free_pernet;
  }
  memset(ptr, 0, mod->core_size);
  mod->module_core = ptr;
@@ -1781,7 +1967,7 @@ static struct module *load_module(void __user *umod,

  /* Fix up syms, so that st_value is a pointer to location. */
  err = simplify_symbols(sechdrs, symindex, strtab, versindex, pcpuindex,
-         mod);
+         pnetindex, mod);
  if (err < 0)
   goto cleanup;

@@ -1860,6 +2046,10 @@ static struct module *load_module(void __user *umod,
  percpu_modcopy(mod->percpu, (void *)sechdrs[pcpuindex].sh_addr,
         sechdrs[pcpuindex].sh_size);

+ /* Copy pernet area over. */
+ pernet_modcopy(mod->pernet, (void *)sechdrs[pnetindex].sh_addr,
+        sechdrs[pnetindex].sh_size);
+
  add_kallsyms(mod, sechdrs, symindex, strindex, secstrings);

  err = module_finalize(hdr, sechdrs, mod);
```

```
@@ -1924,6 +2114,9 @@ static struct module *load_module(void __user *umod,
  cleanup:
  module_unload_free(mod);
  module_free(mod, mod->module_init);
+ free_pernet:
+ if (pernet)
+  pernet_modfree(pernet);
  free_core:
  module_free(mod, mod->module_core);
  free_percpu:
diff --git a/net/core/net_namespace.c b/net/core/net_namespace.c
index 4ae266d..93e3879 100644
--- a/net/core/net_namespace.c
+++ b/net/core/net_namespace.c
@@ -1,4 +1,9 @@
+#include <linux/workqueue.h>
 #include <linux/rtnetlink.h>
+#include <linux/cache.h>
+#include <linux/slab.h>
+#include <linux/list.h>
+#include <linux/delay.h>
 #include <net/net_namespace.h>

 /*
@@ -10,6 +15,233 @@ static struct list_head *first_device = &pernet_list;
 static DEFINE_MUTEX(net_mutex);
 net_t net_template;

+#ifdef CONFIG_NET_NS
+
+static DEFINE_MUTEX(net_list_mutex);
+
+static net_t net_tail;
+static struct kmem_cache *net_cachep;
+static size_t net_size;
+
+/* By using a special section for the first variable in the
+ * per net sectionI get several advantages.
+ * - I can align the entire network namespace structure easily
+ *   to any desired alignment without needing an alignment directive
+ *   in the linker script.  In the worst case the section will start
+ *   with some padding I will never see.
+ * - The code is C so I don't need linker script or header file tricks
+ *   to make the alignment SMP_CACHE_BYTES
+ * - I am guaranteed what the first structure in the network namespace is.
+ *   This allows things like container_of to work and be useful.
+ */
+__attribute__((section(".data.pernet.head"), aligned(SMP_CACHE_BYTES)))
```

```
+struct net_namespace_head __pernetname(net_head) = {
+ .count = ATOMIC_INIT(1),
+ .use_count = ATOMIC_INIT(0),
+};
+EXPORT_PER_NET_SYMBOL_GPL(net_head);
+
+void net_lock(void)
+{
+ mutex_lock(&net_list_mutex);
+}
+
+void net_unlock(void)
+{
+ mutex_unlock(&net_list_mutex);
+}
+
+static void net_list_remove(net_t net)
+{
+ net_t next, prev;
+ BUG_ON(net_eq(net, init_net()));
+
+ next = per_net(net_head, net).next;
+ prev = per_net(net_head, net).prev;
+
+ per_net(net_head, prev).next = next;
+ if (null_net(next)) {
+  net_tail = prev;
+ } else {
+  per_net(net_head, next).prev = prev;
+ }
+}
+
+static void net_list_append(net_t net)
+{
+
+ per_net(net_head, net_tail).next = net;
+ per_net(net_head, net).prev = net_tail;
+ net_tail = net;
+}
+
+static net_t net_alloc(void)
+{
+ return net_of(kmem_cache_alloc(net_cachep, GFP_KERNEL), net_head);
+}
+
+static void net_free(net_t net)
+{
+ struct net_namespace_head *head;
```

```
+ if (null_net(net))
+  return;
+
+ head = &per_net(net_head, net);
+
+ if (unlikely(atomic_read(&head->use_count) != 0)) {
+  printk(KERN_EMERG "network namespace not free! Usage: %d\n",
+   atomic_read(&head->use_count));
+  return;
+ }
+
+ kmem_cache_free(net_cachep, head);
+}
+
+static void cleanup_net(struct work_struct *work)
+{
+ struct pernet_operations *ops;
+ struct list_head *ptr;
+ net_t net;
+
+ net = net_of(work, net_head.work);
+
+ mutex_lock(&net_mutex);
+
+ /* Don't let anyone else find us. */
+ net_lock();
+ net_list_remove(net);
+ net_unlock();
+
+ /* Run all of the network namespace exit methods */
+ list_for_each_prev(ptr, &pernet_list) {
+  ops = list_entry(ptr, struct pernet_operations, list);
+  if (ops->exit)
+   ops->exit(net);
+ }
+
+ mutex_unlock(&net_mutex);
+
+ /* Ensure there are no outstanding rcu callbacks using this
+  * network namespace.
+  */
+ rcu_barrier();
+
+ /* Finally it is safe to free my network namespace structure */
+ net_free(net);
+}
+
+
```

```
+void __put_net(net_t net)
+{
+ /* Cleanup the network namespace in process context */
+ INIT_WORK(&per_net(net_head, net).work, cleanup_net);
+ schedule_work(&per_net(net_head, net).work);
+}
+EXPORT_SYMBOL_GPL(__put_net);
+
+/*
+ * setup_net runs the initializers for the network namespace object.
+ */
+static int setup_net(net_t net)
+{
+ /* Must be called with net_mutex held */
+ struct pernet_operations *ops;
+ struct list_head *ptr;
+ int error;
+
+ /* First initialize the data from the template */
+ memcpy(&per_net(net_head, net), &per_net(net_head, net_template), net_size);
+
+ 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) {
+  ops = list_entry(ptr, struct pernet_operations, list);
+  if (ops->exit)
+   ops->exit(net);
+ }
+ goto out;
+}
+
+void pernet_modcopy(void *pnetdst, const void *src, unsigned long size)
+{
+ net_t net;
+
```

```
+ mutex_lock(&net_mutex);
+ memcpy(pnetdst + __per_net_offset(net_template), src, size);
+ for_each_net(net)
+  memcpy(pnetdst + __per_net_offset(net), src, size);
+ mutex_unlock(&net_mutex);
+}
+
+static int __init net_ns_init(void)
+{
+ size_t init_size;
+ net_t init_net;
+ int err;
+
+ /* Compute the size of the init section */
+ init_size = __per_net_end - __per_net_start;
+
+ /* Compute how large my net namespace structure will be */
+ net_size = ALIGN(init_size, SMP_CACHE_BYTES);
+ net_size += PER_NET_MODULE_RESERVE;
+ net_size = ALIGN(net_size, SMP_CACHE_BYTES);
+
+ printk(KERN_INFO "net_namespace: %zd bytes\n", net_size);
+ net_cachep = kmem_cache_create("net_namespace", net_size,
+     SMP_CACHE_BYTES,
+     SLAB_PANIC, NULL, NULL);
+
+ /* Allocate my template */
+ net_template = net_alloc();
+ if (null_net(net_template))
+  panic("Could not allocate network namespace template");
+
+ /* Initialize my template */
+ memset(&per_net(net_head, net_template), '\0', net_size);
+ memcpy(&per_net(net_head, net_template),
+  &__pernetname(net_head),
+  init_size);
+
+ /* Setup the initial network namespace */
+ init_net = net_alloc();
+ if (null_net(init_net))
+  panic("Could not allocate initial network namespace");
+
+ mutex_lock(&net_mutex);
+ err = setup_net(init_net);
+
+ net_lock();
+ net_tail = init_net;
+ net_unlock();
```

```
+
+ mutex_unlock(&net_mutex);
+ if (err)
+  panic("Could not setup the initial network namespace");
+
+ /* Initialze the init_nsproxy */
+ init_nsproxy.net_ns = init_net;
+
+ return 0;
+}
+
+pure_initcall(net_ns_init);
+
+#endif /* CONFIG_NET_NS */
+
 static int register_pernet_operations(struct list_head *list,
          struct pernet_operations *ops)
 {
--
1.4.4.1.g278f
```

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers