
Subject: [PATCH 7/9] net_device seq_file

Posted by [Andrey Savochkin](#) on Tue, 15 Aug 2006 14:48:47 GMT

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

Library function to create a seq_file in proc filesystem,
showing some information for each netdevice.
This code is present in the kernel in about 10 instances, and
all of them can be converted to using introduced library function.

Signed-off-by: Andrey Savochkin <saw@swsoft.com>

```
include/linux/netdevice.h | 7 +++
net/core/dev.c            | 96 +++++
2 files changed, 103 insertions(+)
```

--- ./include/linux/netdevice.h.venetproc Tue Aug 15 13:46:08 2006

+++ ./include/linux/netdevice.h Tue Aug 15 13:46:08 2006

```
@@ -592,6 +592,13 @@ extern int register_netdevice(struct ne
extern int unregister_netdevice(struct net_device *dev);
extern void free_netdev(struct net_device *dev);
extern void synchronize_net(void);
```

```
+#ifdef CONFIG_PROC_FS
```

```
+extern int netdev_proc_create(char *name,
```

```
+ int (*show)(struct seq_file *,
```

```
+ struct net_device *, void *),
```

```
+ void *data, struct module *mod);
```

```
+void netdev_proc_remove(char *name);
```

```
+#endif
```

```
extern int register_netdevice_notifier(struct notifier_block *nb);
```

```
extern int unregister_netdevice_notifier(struct notifier_block *nb);
```

```
extern int call_netdevice_notifiers(unsigned long val, void *v);
```

--- ./net/core/dev.c.venetproc Tue Aug 15 13:46:08 2006

+++ ./net/core/dev.c Tue Aug 15 13:46:08 2006

```
@@ -2100,6 +2100,102 @@ static int dev_ifconf(char __user *arg)
}
```

```
#ifdef CONFIG_PROC_FS
```

```
+
```

```
+struct netdev_proc_data {
```

```
+ struct file_operations fops;
```

```
+ int (*show)(struct seq_file *, struct net_device *, void *);
```

```
+ void *data;
```

```
+};
```

```
+
```

```
+static void *netdev_proc_seq_start(struct seq_file *seq, loff_t *pos)
```

```
+{
```

```
+ struct net_device *dev;
```

```
+ loff_t off;
```

```

+
+ read_lock(&dev_base_lock);
+ if (*pos == 0)
+ return SEQ_START_TOKEN;
+ for (dev = dev_base, off = 1; dev; dev = dev->next, off++) {
+ if (*pos == off)
+ return dev;
+ }
+ return NULL;
+}
+
+static void *netdev_proc_seq_next(struct seq_file *seq, void *v, loff_t *pos)
+{
+ ++*pos;
+ return (v == SEQ_START_TOKEN) ? dev_base
+ : ((struct net_device *)v)->next;
+}
+
+static void netdev_proc_seq_stop(struct seq_file *seq, void *v)
+{
+ read_unlock(&dev_base_lock);
+}
+
+static int netdev_proc_seq_show(struct seq_file *seq, void *v)
+{
+ struct netdev_proc_data *p;
+
+ p = seq->private;
+ return (*p->show)(seq, v, p->data);
+}
+
+static struct seq_operations netdev_proc_seq_ops = {
+ .start = netdev_proc_seq_start,
+ .next = netdev_proc_seq_next,
+ .stop = netdev_proc_seq_stop,
+ .show = netdev_proc_seq_show,
+};
+
+static int netdev_proc_open(struct inode *inode, struct file *file)
+{
+ int err;
+ struct seq_file *p;
+
+ err = seq_open(file, &netdev_proc_seq_ops);
+ if (!err) {
+ p = file->private_data;
+ p->private = (struct netdev_proc_data *)PDE(inode)->data;
+ }
+}

```

```

+ return err;
+}
+
+int netdev_proc_create(char *name,
+ int (*show)(struct seq_file *, struct net_device *, void *),
+ void *data, struct module *mod)
+{
+ struct netdev_proc_data *p;
+ struct proc_dir_entry *ent;
+
+ p = kzalloc(sizeof(*p), GFP_KERNEL);
+ p->fops.owner = mod;
+ p->fops.open = netdev_proc_open;
+ p->fops.read = seq_read;
+ p->fops.llseek = seq_lseek;
+ p->fops.release = seq_release;
+ p->show = show;
+ p->data = data;
+ ent = create_proc_entry(name, S_IRUGO, proc_net);
+ if (ent == NULL) {
+ kfree(p);
+ return -EINVAL;
+ }
+ ent->data = p;
+ ent->destructor = proc_data_destructor;
+ smp_wmb();
+ ent->proc_fops = &p->fops;
+ return 0;
+}
+EXPORT_SYMBOL(netdev_proc_create);
+
+void netdev_proc_remove(char *name)
+{
+ proc_net_remove(name);
+}
+EXPORT_SYMBOL(netdev_proc_remove);
+
+/*
+ * This is invoked by the /proc filesystem handler to display a device
+ * in detail.

```
