Subject: [PATCH 6/6 net-2.6.25][RAW] Consolidate proc interface (v2)
Posted by Pavel Emelianov on Fri, 16 Nov 2007 15:06:16 GMT
View Forum Message <> Reply to Message

Both ipv6/raw.c and ipv4/raw.c use the seq files to walk
through the raw sockets hash and show them.

The "walking" code is rather huge, but is identical in both
cases. The difference is the hash table to walk over and
the protocol family to check (this was not in the first
virsion of the patch, which was noticed by YOSHIFUJI)

Make the ->open store the needed hash table and the family
on the allocated raw_iter_state and make the start/next/stop
callbacks work with it.

This removes most of the code.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/include/net/raw.h b/include/net/raw.h
index 81a1773..4d1aba0 100644
--- a/include/net/raw.h
+++ b/include/net/raw.h
@@ -37,6 +37,20 @@ struct raw_hashinfo {
 #ifdef CONFIG_PROC_FS
 extern int  raw_proc_init(void);
 extern void raw_proc_exit(void);
+
+struct raw_iter_state {
+ int bucket;
+ unsigned short family;
+ struct raw_hashinfo *h;
+};
+
+#define raw_seq_private(seq) ((struct raw_iter_state *)(seq)->private)
+void *raw_seq_start(struct seq_file *seq, loff_t *pos);
+void *raw_seq_next(struct seq_file *seq, void *v, loff_t *pos);
+void raw_seq_stop(struct seq_file *seq, void *v);
+int raw_seq_open(struct file *file, struct raw_hashinfo *h,
+  unsigned short family);
+
 #endif

 void raw_hash_sk(struct sock *sk, struct raw_hashinfo *h);
diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
```

```
index ffd7a4f..f99828e 100644
--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -843,12 +843,6 @@ struct proto raw_prot = {
 };

 #ifdef CONFIG_PROC_FS
-struct raw_iter_state {
- int bucket;
-};
-
-#define raw_seq_private(seq) ((struct raw_iter_state *)(seq)->private)
-
 static struct sock *raw_get_first(struct seq_file *seq)
 {
  struct sock *sk;
@@ -858,8 +852,8 @@ static struct sock *raw_get_first(struct seq_file *seq)
    ++state->bucket) {
   struct hlist_node *node;

-  sk_for_each(sk, node, &raw_v4_hashinfo.ht[state->bucket])
-   if (sk->sk_family == PF_INET)
+  sk_for_each(sk, node, &state->h->ht[state->bucket])
+   if (sk->sk_family == state->family)
     goto found;
  }
  sk = NULL;
@@ -875,10 +869,10 @@ static struct sock *raw_get_next(struct seq_file *seq, struct sock *sk)
   sk = sk_next(sk);
 try_again:
   ;
-} while (sk && sk->sk_family != PF_INET);
+} while (sk && sk->sk_family != state->family);

  if (!sk && ++state->bucket < RAW_HTABLE_SIZE) {
-  sk = sk_head(&raw_v4_hashinfo.ht[state->bucket]);
+  sk = sk_head(&state->h->ht[state->bucket]);
   goto try_again;
  }
  return sk;
@@ -894,13 +888,16 @@ static struct sock *raw_get_idx(struct seq_file *seq, loff_t pos)
  return pos ? NULL : sk;
 }

-static void *raw_seq_start(struct seq_file *seq, loff_t *pos)
+void *raw_seq_start(struct seq_file *seq, loff_t *pos)
 {
- read_lock(&raw_v4_hashinfo.lock);
```

```
+ struct raw_iter_state *state = raw_seq_private(seq);
+
+ read_lock(&state->h->lock);
  return *pos ? raw_get_idx(seq, *pos - 1) : SEQ_START_TOKEN;
 }
+EXPORT_SYMBOL_GPL(raw_seq_start);

-static void *raw_seq_next(struct seq_file *seq, void *v, loff_t *pos)
+void *raw_seq_next(struct seq_file *seq, void *v, loff_t *pos)
 {
  struct sock *sk;

@@ -911,11 +908,15 @@ static void *raw_seq_next(struct seq_file *seq, void *v, loff_t *pos)
  ++*pos;
  return sk;
 }
+EXPORT_SYMBOL_GPL(raw_seq_next);

-static void raw_seq_stop(struct seq_file *seq, void *v)
+void raw_seq_stop(struct seq_file *seq, void *v)
 {
- read_unlock(&raw_v4_hashinfo.lock);
+ struct raw_iter_state *state = raw_seq_private(seq);
+
+ read_unlock(&state->h->lock);
 }
+EXPORT_SYMBOL_GPL(raw_seq_stop);

 static __inline__ char *get_raw_sock(struct sock *sp, char *tmpbuf, int i)
 {
@@ -962,15 +963,30 @@ static const struct seq_operations raw_seq_ops = {
  .show  = raw_seq_show,
 };

-static int raw_seq_open(struct inode *inode, struct file *file)
+int raw_seq_open(struct file *file, struct raw_hashinfo *h,
+  unsigned short family)
 {
- return seq_open_private(file, &raw_seq_ops,
+ struct raw_iter_state *i;
+
+ i = __seq_open_private(file, &raw_seq_ops,
     sizeof(struct raw_iter_state));
+ if (i == NULL)
+  return -ENOMEM;
+
+ i->h = h;
+ i->family = family;
```

```
+ return 0;
+}
+EXPORT_SYMBOL_GPL(raw_seq_open);
+
+static int raw_v4_seq_open(struct inode *inode, struct file *file)
+{
+ return raw_seq_open(file, &raw_v4_hashinfo, PF_INET);
 }

 static const struct file_operations raw_seq_fops = {
  .owner  = THIS_MODULE,
- .open  = raw_seq_open,
+ .open  = raw_v4_seq_open,
  .read  = seq_read,
  .llseek  = seq_lseek,
  .release = seq_release_private,
diff --git a/net/ipv6/raw.c b/net/ipv6/raw.c
index 422d27c..b34631e 100644
--- a/net/ipv6/raw.c
+++ b/net/ipv6/raw.c
@@ -1200,77 +1200,6 @@ struct proto rawv6_prot = {
 };

 #ifdef CONFIG_PROC_FS
-struct raw6_iter_state {
- int bucket;
-};
-
-#define raw6_seq_private(seq) ((struct raw6_iter_state *)(seq)->private)
-
-static struct sock *raw6_get_first(struct seq_file *seq)
-{
- struct sock *sk;
- struct hlist_node *node;
- struct raw6_iter_state* state = raw6_seq_private(seq);
-
- for (state->bucket = 0; state->bucket < RAW_HTABLE_SIZE;
-   ++state->bucket)
-  sk_for_each(sk, node, &raw_v6_hashinfo.ht[state->bucket])
-   if (sk->sk_family == PF_INET6)
-    goto out;
- sk = NULL;
-out:
- return sk;
-}
-
-static struct sock *raw6_get_next(struct seq_file *seq, struct sock *sk)
-{
```

```
- struct raw6_iter_state* state = raw6_seq_private(seq);
-
- do {
-  sk = sk_next(sk);
-try_again:
-  ;
- } while (sk && sk->sk_family != PF_INET6);
-
- if (!sk && ++state->bucket < RAW_HTABLE_SIZE) {
-  sk = sk_head(&raw_v6_hashinfo.ht[state->bucket]);
-  goto try_again;
- }
- return sk;
-}
-
-static struct sock *raw6_get_idx(struct seq_file *seq, loff_t pos)
-{
- struct sock *sk = raw6_get_first(seq);
- if (sk)
-  while (pos && (sk = raw6_get_next(seq, sk)) != NULL)
-   --pos;
- return pos ? NULL : sk;
-}
-
-static void *raw6_seq_start(struct seq_file *seq, loff_t *pos)
-{
- read_lock(&raw_v6_hashinfo.lock);
- return *pos ? raw6_get_idx(seq, *pos - 1) : SEQ_START_TOKEN;
-}
-
-static void *raw6_seq_next(struct seq_file *seq, void *v, loff_t *pos)
-{
- struct sock *sk;
-
- if (v == SEQ_START_TOKEN)
-  sk = raw6_get_first(seq);
- else
-  sk = raw6_get_next(seq, v);
- ++*pos;
- return sk;
-}
-
-static void raw6_seq_stop(struct seq_file *seq, void *v)
-{
- read_unlock(&raw_v6_hashinfo.lock);
-}
-
 static void raw6_sock_seq_show(struct seq_file *seq, struct sock *sp, int i)
```

```
 {
  struct ipv6_pinfo *np = inet6_sk(sp);
@@ -1308,21 +1237,20 @@ static int raw6_seq_show(struct seq_file *seq, void *v)
       "st tx_queue rx_queue tr tm->when retrnsmt"
       "   uid  timeout inode  drops\n");
  else
-  raw6_sock_seq_show(seq, v, raw6_seq_private(seq)->bucket);
+  raw6_sock_seq_show(seq, v, raw_seq_private(seq)->bucket);
  return 0;
 }

 static const struct seq_operations raw6_seq_ops = {
- .start = raw6_seq_start,
- .next =  raw6_seq_next,
- .stop =  raw6_seq_stop,
+ .start = raw_seq_start,
+ .next =  raw_seq_next,
+ .stop =  raw_seq_stop,
  .show =  raw6_seq_show,
 };

 static int raw6_seq_open(struct inode *inode, struct file *file)
 {
- return seq_open_private(file, &raw6_seq_ops,
-   sizeof(struct raw6_iter_state));
+ return raw_seq_open(file, &raw_v6_hashinfo, PF_INET6);
 }

 static const struct file_operations raw6_seq_fops = {
```