Subject: [PATCH 3/3] sysctl: Error on bad sysctl tables
Posted by ebiederm on Thu, 09 Aug 2007 20:09:29 GMT
View Forum Message <> Reply to Message

After going through the kernels sysctl tables several times it has
become clear that code review and testing is just not effective in
prevent problematic sysctl tables from being used in the stable
kernel. I certainly can't seem to fix the problems as fast as
they are introduced.

Therefore this patch adds sysctl_check_table which is called when a
sysctl table is registered and checks to see if we have a problematic
sysctl table.

The biggest part of the code is the table of valid binary sysctl
entries, but since we have frozen our set of binary sysctls this table
should not need to change, and it makes it much easier to detect
when someone unintentionally adds a new binary sysctl value.

As best as I can determine all of the several hundred errors spewed
on boot up now are legitimate.

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

```
---
 include/linux/sysctl.h |    1 +
 kernel/Makefile        |    3 +-
 kernel/sysctl.c        |    6 +
 kernel/sysctl_check.c  | 1555 ++++++++++++++++++++++++++++++++++++++++++++++++
 4 files changed, 1564 insertions(+), 1 deletions(-)
 create mode 100644 kernel/sysctl_check.c

diff --git a/include/linux/sysctl.h b/include/linux/sysctl.h
index 5ca510b..88f0941 100644
--- a/include/linux/sysctl.h
+++ b/include/linux/sysctl.h
@@ -1048,6 +1048,7 @@ struct ctl_table_header
 struct ctl_table_header *register_sysctl_table(struct ctl_table * table);

 void unregister_sysctl_table(struct ctl_table_header * table);
+int sysctl_check_table(struct ctl_table *table);

 #else /* __KERNEL__ */

diff --git a/kernel/Makefile b/kernel/Makefile
index 2a99983..a8d78ea 100644
--- a/kernel/Makefile
+++ b/kernel/Makefile
@@ -9,8 +9,9 @@ obj-y     = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
```

```
      rcupdate.o extable.o params.o posix-timers.o \
      kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
      hrtimer.o rwsem.o latency.o nsproxy.o srcu.o die_notifier.o \
-     utsname.o
+     utsname.o sysctl.o

+obj-$(CONFIG_SYSCTL) += sysctl_check.o
 obj-$(CONFIG_STACKTRACE) += stacktrace.o
 obj-y += time/
 obj-$(CONFIG_DEBUG_MUTEXES) += mutex-debug.o
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index cf4c632..d6257ee 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -1446,7 +1446,9 @@ static void sysctl_set_parent(struct ctl_table *parent, struct ctl_table
*table)

 static __init int sysctl_init(void)
 {
+ int err;
  sysctl_set_parent(NULL, root_table);
+ err = sysctl_check_table(root_table);
  return 0;
 }

@@ -1531,6 +1533,10 @@ struct ctl_table_header *register_sysctl_table(struct ctl_table * table)
  tmp->used = 0;
  tmp->unregistering = NULL;
  sysctl_set_parent(NULL, table);
+ if (sysctl_check_table(tmp->ctl_table)) {
+  kfree(tmp);
+  return NULL;
+ }
  spin_lock(&sysctl_lock);
  list_add_tail(&tmp->ctl_entry, &root_table_header.ctl_entry);
  spin_unlock(&sysctl_lock);
diff --git a/kernel/sysctl_check.c b/kernel/sysctl_check.c
new file mode 100644
index 0000000..389c4ba
--- /dev/null
+++ b/kernel/sysctl_check.c
@@ -0,0 +1,1555 @@
+#include <linux/stat.h>
+#include <linux/sysctl.h>
+#include "../arch/s390/appldata/appldata.h"
+#include "../fs/xfs/linux-2.6/xfs_sysctl.h"
+#include <linux/sunrpc/debug.h>
+#include <net/ip_vs.h>
```

```
+
+struct trans_ctl_table {
+ int   ctl_name;
+ const char  *procname;
+ struct trans_ctl_table *child;
+};
+
+static struct trans_ctl_table trans_random_table[] = {
+ { RANDOM_POOLSIZE, "poolsize" },
+ { RANDOM_ENTROPY_COUNT, "entropy_avail" },
+ { RANDOM_READ_THRESH, "read_wakeup_threshold" },
+ { RANDOM_WRITE_THRESH, "write_wakeup_threshold" },
+ { RANDOM_BOOT_ID, "boot_id" },
+ { RANDOM_UUID,  "uuid" },
+ {}
+};
+
+static struct trans_ctl_table trans_pty_table[] = {
+ { PTY_MAX,  "max" },
+ { PTY_NR,  "nr" },
+ {}
+};
+
+static struct trans_ctl_table trans_kern_table[] = {
+ { KERN_OSTYPE,   "ostype" },
+ { KERN_OSRELEASE,  "osrelease" },
+ /* KERN_OSREV not used */
+ { KERN_VERSION,   "version" },
+ /* KERN_SECUREMASK not used */
+ /* KERN_PROF not used */
+ { KERN_NODENAME,  "hostname" },
+ { KERN_DOMAINNAME,  "domainname" },
+
+ { KERN_CAP_BSET,  "cap-bound" },
+ { KERN_PANIC,   "panic" },
+ { KERN_REALROOTDEV,  "real-root-dev" },
+
+ { KERN_SPARC_REBOOT,  "reboot-cmd" },
+ { KERN_CTLALTDEL,  "ctrl-alt-del" },
+ { KERN_PRINTK,   "printk" },
+
+ /* KERN_NAMETRANS not used */
+ /* KERN_PPC_HTABRECLAIM not used */
+ /* KERN_PPC_ZEROPAGED not used */
+ { KERN_PPC_POWERSAVE_NAP, "powersave-nap" },
+
+ { KERN_MODPROBE,  "modprobe" },
+ { KERN_SG_BIG_BUFF,  "sg-big-buff" },
```

```
+ { KERN_ACCT,   "acct" },
+ { KERN_PPC_L2CR,  "l2cr" },
+
+ /* KERN_RTSIGNR not used */
+ /* KERN_RTSIGMAX not used */
+
+ { KERN_SHMMAX,   "shmmax" },
+ { KERN_MSGMAX,   "msgmax" },
+ { KERN_MSGMNB,   "msgmnb" },
+ /* KERN_MSGPOOL not used*/
+ { KERN_SYSRQ,   "sysrq" },
+ { KERN_MAX_THREADS,  "threads-max" },
+ { KERN_RANDOM,   "random", trans_random_table },
+ { KERN_SHMALL,   "shmall" },
+ { KERN_MSGMNI,   "msgmni" },
+ { KERN_SEM,   "sem" },
+ { KERN_SPARC_STOP_A,  "stop-a" },
+ { KERN_SHMMNI,   "shmmni" },
+
+ { KERN_OVERFLOWUID,  "overflowuid" },
+ { KERN_OVERFLOWGID,  "overflowgid" },
+
+ { KERN_HOTPLUG,   "hotplug", },
+ { KERN_IEEE_EMULATION_WARNINGS, "ieee_emulation_warnings" },
+
+ { KERN_S390_USER_DEBUG_LOGGING, "userprocess_debug" },
+ { KERN_CORE_USES_PID,  "core_uses_pid" },
+ { KERN_TAINTED,   "tainted" },
+ { KERN_CADPID,   "cad_pid" },
+ { KERN_PIDMAX,   "pid_max" },
+ { KERN_CORE_PATTERN,  "core_pattern" },
+ { KERN_PANIC_ON_OOPS,  "panic_on_oops" },
+ { KERN_HPPA_PWRSW,  "soft-power" },
+ { KERN_HPPA_UNALIGNED,  "unaligned-trap" },
+
+ { KERN_PRINTK_RATELIMIT, "printk_ratelimit" },
+ { KERN_PRINTK_RATELIMIT_BURST, "printk_ratelimit_burst" },
+
+ { KERN_PTY,   "pty",  trans_pty_table },
+ { KERN_NGROUPS_MAX,  "ngroups_max" },
+ { KERN_SPARC_SCONS_PWROFF, "scons_poweroff" },
+ { KERN_HZ_TIMER,  "hz_timer" },
+ { KERN_UNKNOWN_NMI_PANIC, "unknown_nmi_panic" },
+ { KERN_BOOTLOADER_TYPE,  "bootloader_type" },
+ { KERN_RANDOMIZE,  "randomize_va_space" },
+
+ { KERN_SPIN_RETRY,  "spin_retry" },
+ { KERN_ACPI_VIDEO_FLAGS, "acpi_video_flags" },
```

```
+ { KERN_IA64_UNALIGNED,  "ignore-unaligned-usertrap" },
+ { KERN_COMPAT_LOG,  "compat-log" },
+ { KERN_MAX_LOCK_DEPTH,  "max_lock_depth" },
+ { KERN_NMI_WATCHDOG,  "nmi_watchdog" },
+ { KERN_PANIC_ON_NMI,  "panic_on_unrecovered_nmi" },
+ {}
+};
+
+static struct trans_ctl_table trans_vm_table[] = {
+ { VM_OVERCOMMIT_MEMORY,  "overcommit_memory" },
+ { VM_PAGE_CLUSTER,  "page-cluster" },
+ { VM_DIRTY_BACKGROUND,  "dirty_background_ratio" },
+ { VM_DIRTY_RATIO,  "dirty_ratio" },
+ { VM_DIRTY_WB_CS,  "dirty_writeback_centisecs" },
+ { VM_DIRTY_EXPIRE_CS,  "dirty_expire_centisecs" },
+ { VM_NR_PDFLUSH_THREADS, "nr_pdflush_threads" },
+ { VM_OVERCOMMIT_RATIO,  "overcommit_ratio" },
+ /* VM_PAGEBUF unused */
+ { VM_HUGETLB_PAGES,  "nr_hugepages" },
+ { VM_SWAPPINESS,  "swappiness" },
+ { VM_LOWMEM_RESERVE_RATIO, "lowmem_reserve_ratio" },
+ { VM_MIN_FREE_KBYTES,  "min_free_kbytes" },
+ { VM_MAX_MAP_COUNT,  "max_map_count" },
+ { VM_LAPTOP_MODE,  "laptop_mode" },
+ { VM_BLOCK_DUMP,  "block_dump" },
+ { VM_HUGETLB_GROUP,  "hugetlb_shm_group" },
+ { VM_VFS_CACHE_PRESSURE, "vfs_cache_pressure" },
+ { VM_LEGACY_VA_LAYOUT,  "legacy_va_layout" },
+ /* VM_SWAP_TOKEN_TIMEOUT unused */
+ { VM_DROP_PAGECACHE,  "drop_caches" },
+ { VM_PERCPU_PAGELIST_FRACTION, "percpu_pagelist_fraction" },
+ { VM_ZONE_RECLAIM_MODE,  "zone_reclaim_mode" },
+ { VM_MIN_UNMAPPED,  "min_unmapped_ratio" },
+ { VM_PANIC_ON_OOM,  "panic_on_oom" },
+ { VM_VDSO_ENABLED,  "vdso_enabled" },
+ { VM_MIN_SLAB,   "min_slab_ratio" },
+ { VM_CMM_PAGES,   "cmm_pages" },
+ { VM_CMM_TIMED_PAGES,  "cmm_timed_pages" },
+ { VM_CMM_TIMEOUT,  "cmm_timeout" },
+
+ {}
+};
+
+static struct trans_ctl_table trans_net_core_table[] = {
+ { NET_CORE_WMEM_MAX,  "wmem_max" },
+ { NET_CORE_RMEM_MAX,  "rmem_max" },
+ { NET_CORE_WMEM_DEFAULT, "wmem_default" },
+ { NET_CORE_RMEM_DEFAULT, "rmem_default" },
```

```
+ /* NET_CORE_DESTROY_DELAY unused */
+ { NET_CORE_MAX_BACKLOG,  "netdev_max_backlog" },
+ /* NET_CORE_FASTROUTE unused */
+ { NET_CORE_MSG_COST,  "message_cost" },
+ { NET_CORE_MSG_BURST,  "message_burst" },
+ { NET_CORE_OPTMEM_MAX,  "optmem_max" },
+ /* NET_CORE_HOT_LIST_LENGTH unused */
+ /* NET_CORE_DIVERT_VERSION unused */
+ /* NET_CORE_NO_CONG_THRESH unused */
+ /* NET_CORE_NO_CONG unused */
+ /* NET_CORE_LO_CONG unused */
+ /* NET_CORE_MOD_CONG unused */
+ { NET_CORE_DEV_WEIGHT,  "dev_weight" },
+ { NET_CORE_SOMAXCONN,  "somaxconn" },
+ { NET_CORE_BUDGET,  "netdev_budget" },
+ { NET_CORE_AEVENT_ETIME, "xfrm_aevent_etime" },
+ { NET_CORE_AEVENT_RSEQTH, "xfrm_aevent_rseqth" },
+ { NET_CORE_WARNINGS,  "warnings" },
+ {},
+};
+
+static struct trans_ctl_table trans_net_unix_table[] = {
+ /* NET_UNIX_DESTROY_DELAY unused */
+ /* NET_UNIX_DELETE_DELAY unused */
+ { NET_UNIX_MAX_DGRAM_QLEN, "max_dgram_qlen" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv4_route_table[] = {
+ { NET_IPV4_ROUTE_FLUSH,   "flush" },
+ { NET_IPV4_ROUTE_MIN_DELAY,  "min_delay" },
+ { NET_IPV4_ROUTE_MAX_DELAY,  "max_delay" },
+ { NET_IPV4_ROUTE_GC_THRESH,  "gc_thresh" },
+ { NET_IPV4_ROUTE_MAX_SIZE,  "max_size" },
+ { NET_IPV4_ROUTE_GC_MIN_INTERVAL, "gc_min_interval" },
+ { NET_IPV4_ROUTE_GC_TIMEOUT,  "gc_timeout" },
+ { NET_IPV4_ROUTE_GC_INTERVAL,  "gc_interval" },
+ { NET_IPV4_ROUTE_REDIRECT_LOAD,  "redirect_load" },
+ { NET_IPV4_ROUTE_REDIRECT_NUMBER, "redirect_number" },
+ { NET_IPV4_ROUTE_REDIRECT_SILENCE, "redirect_silence" },
+ { NET_IPV4_ROUTE_ERROR_COST,  "error_cost" },
+ { NET_IPV4_ROUTE_ERROR_BURST,  "error_burst" },
+ { NET_IPV4_ROUTE_GC_ELASTICITY,  "gc_elasticity" },
+ { NET_IPV4_ROUTE_MTU_EXPIRES,  "mtu_expires" },
+ { NET_IPV4_ROUTE_MIN_PMTU,  "min_pmtu" },
+ { NET_IPV4_ROUTE_MIN_ADVMSS,  "min_adv_mss" },
+ { NET_IPV4_ROUTE_SECRET_INTERVAL, "secret_interval" },
+ { NET_IPV4_ROUTE_GC_MIN_INTERVAL_MS, "gc_min_interval_ms" },
```

```
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv4_conf_vars_table[] = {
+ { NET_IPV4_CONF_FORWARDING,  "forwarding" },
+ { NET_IPV4_CONF_MC_FORWARDING,  "mc_forwarding" },
+
+ { NET_IPV4_CONF_PROXY_ARP,  "proxy_arp" },
+ { NET_IPV4_CONF_ACCEPT_REDIRECTS, "accept_redirects" },
+ { NET_IPV4_CONF_SECURE_REDIRECTS, "secure_redirects" },
+ { NET_IPV4_CONF_SEND_REDIRECTS,  "send_redirects" },
+ { NET_IPV4_CONF_SHARED_MEDIA,  "shared_media" },
+ { NET_IPV4_CONF_RP_FILTER,  "rp_filter" },
+ { NET_IPV4_CONF_ACCEPT_SOURCE_ROUTE, "accept_source_route" },
+ { NET_IPV4_CONF_BOOTP_RELAY,  "bootp_relay" },
+ { NET_IPV4_CONF_LOG_MARTIANS,  "log_martians" },
+ { NET_IPV4_CONF_TAG,   "tag" },
+ { NET_IPV4_CONF_ARPFILTER,  "arp_filter" },
+ { NET_IPV4_CONF_MEDIUM_ID,  "medium_id" },
+ { NET_IPV4_CONF_NOXFRM,   "disable_xfrm" },
+ { NET_IPV4_CONF_NOPOLICY,  "disable_policy" },
+ { NET_IPV4_CONF_FORCE_IGMP_VERSION, "force_igmp_version" },
+
+ { NET_IPV4_CONF_ARP_ANNOUNCE,  "arp_announce" },
+ { NET_IPV4_CONF_ARP_IGNORE,  "arp_ignore" },
+ { NET_IPV4_CONF_PROMOTE_SECONDARIES, "promote_secondaries" },
+ { NET_IPV4_CONF_ARP_ACCEPT,  "arp_accept" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv4_conf_table[] = {
+ { NET_PROTO_CONF_ALL,  "all",  trans_net_ipv4_conf_vars_table },
+ { NET_PROTO_CONF_DEFAULT, "default", trans_net_ipv4_conf_vars_table },
+ { 0, NULL, trans_net_ipv4_conf_vars_table },
+ {}
+};
+
+
+static struct trans_ctl_table trans_net_ipv4_vs_table[] = {
+ { NET_IPV4_VS_AMEMTHRESH, "amemthresh" },
+ { NET_IPV4_VS_DEBUG_LEVEL, "debug_level" },
+ { NET_IPV4_VS_AMDROPRATE, "am_droprate" },
+ { NET_IPV4_VS_DROP_ENTRY, "drop_entry" },
+ { NET_IPV4_VS_DROP_PACKET, "drop_packet" },
+ { NET_IPV4_VS_SECURE_TCP, "secure_tcp" },
+ { NET_IPV4_VS_TO_ES,  "timeout_established" },
+ { NET_IPV4_VS_TO_SS,  "timeout_synsent" },
+ { NET_IPV4_VS_TO_SR,  "timeout_synrecv" },
```

```
+ { NET_IPV4_VS_TO_FW,  "timeout_finwait" },
+ { NET_IPV4_VS_TO_TW,  "timeout_timewait" },
+ { NET_IPV4_VS_TO_CL,  "timeout_close" },
+ { NET_IPV4_VS_TO_CW,  "timeout_closewait" },
+ { NET_IPV4_VS_TO_LA,  "timeout_lastack" },
+ { NET_IPV4_VS_TO_LI,  "timeout_listen" },
+ { NET_IPV4_VS_TO_SA,  "timeout_synack" },
+ { NET_IPV4_VS_TO_UDP,  "timeout_udp" },
+ { NET_IPV4_VS_TO_ICMP,  "timeout_icmp" },
+ { NET_IPV4_VS_CACHE_BYPASS, "cache_bypass" },
+ { NET_IPV4_VS_EXPIRE_NODEST_CONN, "expire_nodest_conn" },
+ { NET_IPV4_VS_EXPIRE_QUIESCENT_TEMPLATE, "expire_quiescent_template" },
+ { NET_IPV4_VS_SYNC_THRESHOLD,  "sync_threshold" },
+ { NET_IPV4_VS_NAT_ICMP_SEND, "nat_icmp_send" },
+ { NET_IPV4_VS_LBLC_EXPIRE,  "lblc_expiration" },
+ { NET_IPV4_VS_LBLCR_EXPIRE,  "lblcr_expiration" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_neigh_vars_table[] = {
+ { NET_NEIGH_MCAST_SOLICIT, "mcast_solicit" },
+ { NET_NEIGH_UCAST_SOLICIT, "ucast_solicit" },
+ { NET_NEIGH_APP_SOLICIT, "app_solicit" },
+ { NET_NEIGH_RETRANS_TIME, "retrans_time" },
+ { NET_NEIGH_REACHABLE_TIME, "base_reachable_time" },
+ { NET_NEIGH_DELAY_PROBE_TIME, "delay_first_probe_time" },
+ { NET_NEIGH_GC_STALE_TIME, "gc_stale_time" },
+ { NET_NEIGH_UNRES_QLEN,  "unres_qlen" },
+ { NET_NEIGH_PROXY_QLEN,  "proxy_qlen" },
+ { NET_NEIGH_ANYCAST_DELAY, "anycast_delay" },
+ { NET_NEIGH_PROXY_DELAY, "proxy_delay" },
+ { NET_NEIGH_LOCKTIME,  "locktime" },
+ { NET_NEIGH_GC_INTERVAL, "gc_interval" },
+ { NET_NEIGH_GC_THRESH1,  "gc_thresh1" },
+ { NET_NEIGH_GC_THRESH2,  "gc_thresh2" },
+ { NET_NEIGH_GC_THRESH3,  "gc_thresh3" },
+ { NET_NEIGH_RETRANS_TIME_MS, "retrans_time_ms" },
+ { NET_NEIGH_REACHABLE_TIME_MS, "base_reachable_time_ms" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_neigh_table[] = {
+ { NET_PROTO_CONF_DEFAULT, "default", trans_net_neigh_vars_table },
+ { 0, NULL, trans_net_neigh_vars_table },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv4_netfilter_table[] = {
```

```
+ { NET_IPV4_NF_CONNTRACK_MAX,    "ip_conntrack_max" },
+
+ {
NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_SYN_SENT,  "ip_conntrack_tcp_timeout_syn_se
nt" },
+ {
NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_SYN_RECV,  "ip_conntrack_tcp_timeout_syn_re
cv" },
+ {
NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_ESTABLISHED, "ip_conntrack_tcp_timeout_esta
blished" },
+ {
NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_FIN_WAIT,  "ip_conntrack_tcp_timeout_fin_wait"
},
+ {
NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_CLOSE_WAIT,  "ip_conntrack_tcp_timeout_clos
e_wait" },
+ {
NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_LAST_ACK,  "ip_conntrack_tcp_timeout_last_ac
k" },
+ {
NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_TIME_WAIT,  "ip_conntrack_tcp_timeout_time_w
ait" },
+ { NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_CLOSE,  "ip_conntrack_tcp_timeout_close" },
+
+ { NET_IPV4_NF_CONNTRACK_UDP_TIMEOUT,   "ip_conntrack_udp_timeout" },
+ {
NET_IPV4_NF_CONNTRACK_UDP_TIMEOUT_STREAM,  "ip_conntrack_udp_timeout_stream"
},
+ { NET_IPV4_NF_CONNTRACK_ICMP_TIMEOUT,   "ip_conntrack_icmp_timeout" },
+ { NET_IPV4_NF_CONNTRACK_GENERIC_TIMEOUT,  "ip_conntrack_generic_timeout" },
+
+ { NET_IPV4_NF_CONNTRACK_BUCKETS,   "ip_conntrack_buckets" },
+ { NET_IPV4_NF_CONNTRACK_LOG_INVALID,   "ip_conntrack_log_invalid" },
+ {
NET_IPV4_NF_CONNTRACK_TCP_TIMEOUT_MAX_RETRANS, "ip_conntrack_tcp_timeout_ma
x_retrans" },
+ { NET_IPV4_NF_CONNTRACK_TCP_LOOSE,   "ip_conntrack_tcp_loose" },
+ { NET_IPV4_NF_CONNTRACK_TCP_BE_LIBERAL,   "ip_conntrack_tcp_be_liberal" },
+ { NET_IPV4_NF_CONNTRACK_TCP_MAX_RETRANS,  "ip_conntrack_tcp_max_retrans" },
+
+ {
NET_IPV4_NF_CONNTRACK_SCTP_TIMEOUT_CLOSED,  "ip_conntrack_sctp_timeout_closed"
},
+ {
NET_IPV4_NF_CONNTRACK_SCTP_TIMEOUT_COOKIE_WAIT, "ip_conntrack_sctp_timeout_c
ookie_wait" },
+ {
```

```
NET_IPV4_NF_CONNTRACK_SCTP_TIMEOUT_COOKIE_ECHOED, "ip_conntrack_sctp_timeo
ut_cookie_echoed" },
+ {
NET_IPV4_NF_CONNTRACK_SCTP_TIMEOUT_ESTABLISHED, "ip_conntrack_sctp_timeout_e
stablished" },
+ {
NET_IPV4_NF_CONNTRACK_SCTP_TIMEOUT_SHUTDOWN_SENT, "ip_conntrack_sctp_timeo
ut_shutdown_sent" },
+ {
NET_IPV4_NF_CONNTRACK_SCTP_TIMEOUT_SHUTDOWN_RECD, "ip_conntrack_sctp_time
out_shutdown_recd" },
+ {
NET_IPV4_NF_CONNTRACK_SCTP_TIMEOUT_SHUTDOWN_ACK_SENT, "ip_conntrack_sctp
_timeout_shutdown_ack_sent" },
+
+ { NET_IPV4_NF_CONNTRACK_COUNT,  "ip_conntrack_count" },
+ { NET_IPV4_NF_CONNTRACK_CHECKSUM, "ip_conntrack_checksum" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv4_table[] = {
+ { NET_IPV4_FORWARD,   "ip_forward" },
+ { NET_IPV4_DYNADDR,   "ip_dynaddr" },
+
+ { NET_IPV4_CONF,  "conf",  trans_net_ipv4_conf_table },
+ { NET_IPV4_NEIGH,  "neigh", trans_net_neigh_table },
+ { NET_IPV4_ROUTE,  "route", trans_net_ipv4_route_table },
+ /* NET_IPV4_FIB_HASH unused */
+ { NET_IPV4_NETFILTER,  "netfilter", trans_net_ipv4_netfilter_table },
+ { NET_IPV4_VS,   "vs",  trans_net_ipv4_vs_table },
+
+ { NET_IPV4_TCP_TIMESTAMPS,  "tcp_timestamps" },
+ { NET_IPV4_TCP_WINDOW_SCALING,  "tcp_window_scaling" },
+ { NET_IPV4_TCP_SACK,   "tcp_sack" },
+ { NET_IPV4_TCP_RETRANS_COLLAPSE, "tcp_retrans_collapse" },
+ { NET_IPV4_DEFAULT_TTL,   "ip_default_ttl" },
+ /* NET_IPV4_AUTOCONFIG unused */
+ { NET_IPV4_NO_PMTU_DISC,  "ip_no_pmtu_disc" },
+ { NET_IPV4_TCP_SYN_RETRIES,  "tcp_syn_retries" },
+ { NET_IPV4_IPFRAG_HIGH_THRESH,  "ipfrag_high_thresh" },
+ { NET_IPV4_IPFRAG_LOW_THRESH,  "ipfrag_low_thresh" },
+ { NET_IPV4_IPFRAG_TIME,   "ipfrag_time" },
+ /* NET_IPV4_TCP_MAX_KA_PROBES unused */
+ { NET_IPV4_TCP_KEEPALIVE_TIME,  "tcp_keepalive_time" },
+ { NET_IPV4_TCP_KEEPALIVE_PROBES, "tcp_keepalive_probes" },
+ { NET_IPV4_TCP_RETRIES1,  "tcp_retries1" },
+ { NET_IPV4_TCP_RETRIES2,  "tcp_retries2" },
+ { NET_IPV4_TCP_FIN_TIMEOUT,  "tcp_fin_timeout" },
```

```
+ /* NET_IPV4_IP_MASQ_DEBUG unused */
+ { NET_TCP_SYNCOOKIES,   "tcp_syncookies" },
+ { NET_TCP_STDURG,   "tcp_stdurg" },
+ { NET_TCP_RFC1337,   "tcp_rfc1337" },
+ /* NET_TCP_SYN_TAILDROP unused */
+ { NET_TCP_MAX_SYN_BACKLOG,  "tcp_max_syn_backlog" },
+ { NET_IPV4_LOCAL_PORT_RANGE,  "ip_local_port_range" },
+ { NET_IPV4_ICMP_ECHO_IGNORE_ALL, "icmp_echo_ignore_all" },
+ { NET_IPV4_ICMP_ECHO_IGNORE_BROADCASTS, "icmp_echo_ignore_broadcasts" },
+ /* NET_IPV4_ICMP_SOURCEQUENCH_RATE unused */
+ /* NET_IPV4_ICMP_DESTUNREACH_RATE unused */
+ /* NET_IPV4_ICMP_TIMEEXCEED_RATE unused */
+ /* NET_IPV4_ICMP_PARAMPROB_RATE unused */
+ /* NET_IPV4_ICMP_ECHOREPLY_RATE unused */
+ {
NET_IPV4_ICMP_IGNORE_BOGUS_ERROR_RESPONSES, "icmp_ignore_bogus_error_respon
ses" },
+ { NET_IPV4_IGMP_MAX_MEMBERSHIPS, "igmp_max_memberships" },
+ { NET_TCP_TW_RECYCLE,   "tcp_tw_recycle" },
+ /* NET_IPV4_ALWAYS_DEFRAG unused */
+ { NET_IPV4_TCP_KEEPALIVE_INTVL,  "tcp_keepalive_intvl" },
+ { NET_IPV4_INET_PEER_THRESHOLD,  "inet_peer_threshold" },
+ { NET_IPV4_INET_PEER_MINTTL,  "inet_peer_minttl" },
+ { NET_IPV4_INET_PEER_MAXTTL,  "inet_peer_maxttl" },
+ { NET_IPV4_INET_PEER_GC_MINTIME, "inet_peer_gc_mintime" },
+ { NET_IPV4_INET_PEER_GC_MAXTIME, "inet_peer_gc_maxtime" },
+ { NET_TCP_ORPHAN_RETRIES,  "tcp_orphan_retries" },
+ { NET_TCP_ABORT_ON_OVERFLOW,  "tcp_abort_on_overflow" },
+ { NET_TCP_SYNACK_RETRIES,  "tcp_synack_retries" },
+ { NET_TCP_MAX_ORPHANS,   "tcp_max_orphans" },
+ { NET_TCP_MAX_TW_BUCKETS,  "tcp_max_tw_buckets" },
+ { NET_TCP_FACK,    "tcp_fack" },
+ { NET_TCP_REORDERING,   "tcp_reordering" },
+ { NET_TCP_ECN,    "tcp_ecn" },
+ { NET_TCP_DSACK,   "tcp_dsack" },
+ { NET_TCP_MEM,    "tcp_mem" },
+ { NET_TCP_WMEM,    "tcp_wmem" },
+ { NET_TCP_RMEM,    "tcp_rmem" },
+ { NET_TCP_APP_WIN,   "tcp_app_win" },
+ { NET_TCP_ADV_WIN_SCALE,  "tcp_adv_win_scale" },
+ { NET_IPV4_NONLOCAL_BIND,  "ip_nonlocal_bind" },
+ { NET_IPV4_ICMP_RATELIMIT,  "icmp_ratelimit" },
+ { NET_IPV4_ICMP_RATEMASK,  "icmp_ratemask" },
+ { NET_TCP_TW_REUSE,   "tcp_tw_reuse" },
+ { NET_TCP_FRTO,    "tcp_frto" },
+ { NET_TCP_LOW_LATENCY,   "tcp_low_latency" },
+ { NET_IPV4_IPFRAG_SECRET_INTERVAL, "ipfrag_secret_interval" },
+ { NET_IPV4_IGMP_MAX_MSF,  "igmp_max_msf" },
```

```
+ { NET_TCP_NO_METRICS_SAVE,  "tcp_no_metrics_save" },
+ /* NET_TCP_DEFAULT_WIN_SCALE unused */
+ { NET_TCP_MODERATE_RCVBUF,  "tcp_moderate_rcvbuf" },
+ { NET_TCP_TSO_WIN_DIVISOR,  "tcp_tso_win_divisor" },
+ /* NET_TCP_BIC_BETA unused */
+ { NET_IPV4_ICMP_ERRORS_USE_INBOUND_IFADDR, "icmp_errors_use_inbound_ifaddr" },
+ { NET_TCP_CONG_CONTROL,   "tcp_congestion_control" },
+ { NET_TCP_ABC,    "tcp_abc" },
+ { NET_IPV4_IPFRAG_MAX_DIST,  "ipfrag_max_dist" },
+ { NET_TCP_MTU_PROBING,   "tcp_mtu_probing" },
+ { NET_TCP_BASE_MSS,   "tcp_base_mss" },
+ { NET_IPV4_TCP_WORKAROUND_SIGNED_WINDOWS, "tcp_workaround_signed_windows"
},
+ { NET_TCP_DMA_COPYBREAK,  "tcp_dma_copybreak" },
+ { NET_TCP_SLOW_START_AFTER_IDLE, "tcp_slow_start_after_idle" },
+ { NET_CIPSOV4_CACHE_ENABLE,  "cipso_cache_enable" },
+ { NET_CIPSOV4_CACHE_BUCKET_SIZE, "cipso_cache_bucket_size" },
+ { NET_CIPSOV4_RBM_OPTFMT,  "cipso_rbm_optfmt" },
+ { NET_CIPSOV4_RBM_STRICTVALID,  "cipso_rbm_strictvalid" },
+ { NET_TCP_AVAIL_CONG_CONTROL,  "tcp_available_congestion_control" },
+ { NET_TCP_ALLOWED_CONG_CONTROL,  "tcp_allowed_congestion_control" },
+ { NET_TCP_MAX_SSTHRESH,   "tcp_max_ssthresh" },
+ { NET_TCP_FRTO_RESPONSE,  "tcp_frto_response" },
+ { 2088 /* NET_IPQ_QMAX */,  "ip_queue_maxlen" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipx_table[] = {
+ { NET_IPX_PPROP_BROADCASTING, "ipx_pprop_broadcasting" },
+ /* NET_IPX_FORWARDING unused */
+ {}
+};
+
+static struct trans_ctl_table trans_net_atalk_table[] = {
+ { NET_ATALK_AARP_EXPIRY_TIME,  "aarp-expiry-time" },
+ { NET_ATALK_AARP_TICK_TIME,  "aarp-tick-time" },
+ { NET_ATALK_AARP_RETRANSMIT_LIMIT, "aarp-retransmit-limit" },
+ { NET_ATALK_AARP_RESOLVE_TIME,  "aarp-resolve-time" },
+ {},
+};
+
+static struct trans_ctl_table trans_net_netrom_table[] = {
+ { NET_NETROM_DEFAULT_PATH_QUALITY,  "default_path_quality" },
+ { NET_NETROM_OBSOLESCENCE_COUNT_INITIALISER, "obsolescence_count_initialiser" },
+ { NET_NETROM_NETWORK_TTL_INITIALISER,  "network_ttl_initialiser" },
+ { NET_NETROM_TRANSPORT_TIMEOUT,   "transport_timeout" },
+ { NET_NETROM_TRANSPORT_MAXIMUM_TRIES,  "transport_maximum_tries" },
+ { NET_NETROM_TRANSPORT_ACKNOWLEDGE_DELAY, "transport_acknowledge_delay" },
```

```
+ { NET_NETROM_TRANSPORT_BUSY_DELAY,  "transport_busy_delay" },
+ {
NET_NETROM_TRANSPORT_REQUESTED_WINDOW_SIZE, "transport_requested_window_si
ze" },
+ { NET_NETROM_TRANSPORT_NO_ACTIVITY_TIMEOUT, "transport_no_activity_timeout" },
+ { NET_NETROM_ROUTING_CONTROL,   "routing_control" },
+ { NET_NETROM_LINK_FAILS_COUNT,   "link_fails_count" },
+ { NET_NETROM_RESET,    "reset" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ax25_table[] = {
+ { NET_AX25_IP_DEFAULT_MODE, "ip_default_mode" },
+ { NET_AX25_DEFAULT_MODE, "ax25_default_mode" },
+ { NET_AX25_BACKOFF_TYPE, "backoff_type" },
+ { NET_AX25_CONNECT_MODE, "connect_mode" },
+ { NET_AX25_STANDARD_WINDOW, "standard_window_size" },
+ { NET_AX25_EXTENDED_WINDOW, "extended_window_size" },
+ { NET_AX25_T1_TIMEOUT,  "t1_timeout" },
+ { NET_AX25_T2_TIMEOUT,  "t2_timeout" },
+ { NET_AX25_T3_TIMEOUT,  "t3_timeout" },
+ { NET_AX25_IDLE_TIMEOUT, "idle_timeout" },
+ { NET_AX25_N2,   "maximum_retry_count" },
+ { NET_AX25_PACLEN,  "maximum_packet_length" },
+ { NET_AX25_PROTOCOL,  "protocol" },
+ { NET_AX25_DAMA_SLAVE_TIMEOUT, "dama_slave_timeout" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_bridge_table[] = {
+ { NET_BRIDGE_NF_CALL_ARPTABLES,  "bridge-nf-call-arptables" },
+ { NET_BRIDGE_NF_CALL_IPTABLES,  "bridge-nf-call-iptables" },
+ { NET_BRIDGE_NF_CALL_IP6TABLES,  "bridge-nf-call-ip6tables" },
+ { NET_BRIDGE_NF_FILTER_VLAN_TAGGED, "bridge-nf-filter-vlan-tagged" },
+ { NET_BRIDGE_NF_FILTER_PPPOE_TAGGED, "bridge-nf-filter-pppoe-tagged" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_rose_table[] = {
+ { NET_ROSE_RESTART_REQUEST_TIMEOUT, "restart_request_timeout" },
+ { NET_ROSE_CALL_REQUEST_TIMEOUT, "call_request_timeout" },
+ { NET_ROSE_RESET_REQUEST_TIMEOUT, "reset_request_timeout" },
+ { NET_ROSE_CLEAR_REQUEST_TIMEOUT, "clear_request_timeout" },
+ { NET_ROSE_ACK_HOLD_BACK_TIMEOUT, "acknowledge_hold_back_timeout" },
+ { NET_ROSE_ROUTING_CONTROL,  "routing_control" },
+ { NET_ROSE_LINK_FAIL_TIMEOUT,  "link_fail_timeout" },
+ { NET_ROSE_MAX_VCS,   "maximum_virtual_circuits" },
+ { NET_ROSE_WINDOW_SIZE,   "window_size" },
```

```
+ { NET_ROSE_NO_ACTIVITY_TIMEOUT,  "no_activity_timeout" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv6_conf_var_table[] = {
+ { NET_IPV6_FORWARDING,   "forwarding" },
+ { NET_IPV6_HOP_LIMIT,   "hop_limit" },
+ { NET_IPV6_MTU,    "mtu" },
+ { NET_IPV6_ACCEPT_RA,   "accept_ra" },
+ { NET_IPV6_ACCEPT_REDIRECTS,  "accept_redirects" },
+ { NET_IPV6_AUTOCONF,   "autoconf" },
+ { NET_IPV6_DAD_TRANSMITS,  "dad_transmits" },
+ { NET_IPV6_RTR_SOLICITS,  "router_solicitations" },
+ { NET_IPV6_RTR_SOLICIT_INTERVAL, "router_solicitation_interval" },
+ { NET_IPV6_RTR_SOLICIT_DELAY,  "router_solicitation_delay" },
+ { NET_IPV6_USE_TEMPADDR,  "use_tempaddr" },
+ { NET_IPV6_TEMP_VALID_LFT,  "temp_valid_lft" },
+ { NET_IPV6_TEMP_PREFERED_LFT,  "temp_prefered_lft" },
+ { NET_IPV6_REGEN_MAX_RETRY,  "regen_max_retry" },
+ { NET_IPV6_MAX_DESYNC_FACTOR,  "max_desync_factor" },
+ { NET_IPV6_MAX_ADDRESSES,  "max_addresses" },
+ { NET_IPV6_FORCE_MLD_VERSION,  "force_mld_version" },
+ { NET_IPV6_ACCEPT_RA_DEFRTR,  "accept_ra_defrtr" },
+ { NET_IPV6_ACCEPT_RA_PINFO,  "accept_ra_pinfo" },
+ { NET_IPV6_ACCEPT_RA_RTR_PREF,  "accept_ra_rtr_pref" },
+ { NET_IPV6_RTR_PROBE_INTERVAL,  "router_probe_interval" },
+ { NET_IPV6_ACCEPT_RA_RT_INFO_MAX_PLEN, "accept_ra_rt_info_max_plen" },
+ { NET_IPV6_PROXY_NDP,   "proxy_ndp" },
+ { NET_IPV6_ACCEPT_SOURCE_ROUTE,  "accept_source_route" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv6_conf_table[] = {
+ { NET_PROTO_CONF_ALL,  "all", trans_net_ipv6_conf_var_table },
+ { NET_PROTO_CONF_DEFAULT,  "default", trans_net_ipv6_conf_var_table },
+ { 0, NULL, trans_net_ipv6_conf_var_table },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv6_route_table[] = {
+ { NET_IPV6_ROUTE_FLUSH,   "flush" },
+ { NET_IPV6_ROUTE_GC_THRESH,  "gc_thresh" },
+ { NET_IPV6_ROUTE_MAX_SIZE,  "max_size" },
+ { NET_IPV6_ROUTE_GC_MIN_INTERVAL, "gc_min_interval" },
+ { NET_IPV6_ROUTE_GC_TIMEOUT,  "gc_timeout" },
+ { NET_IPV6_ROUTE_GC_INTERVAL,  "gc_interval" },
+ { NET_IPV6_ROUTE_GC_ELASTICITY,  "gc_elasticity" },
+ { NET_IPV6_ROUTE_MTU_EXPIRES,  "mtu_expires" },
```

```
+ { NET_IPV6_ROUTE_MIN_ADVMSS,  "min_adv_mss" },
+ { NET_IPV6_ROUTE_GC_MIN_INTERVAL_MS, "gc_min_interval_ms" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv6_icmp_table[] = {
+ { NET_IPV6_ICMP_RATELIMIT, "ratelimit" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_ipv6_table[] = {
+ { NET_IPV6_CONF,  "conf",  trans_net_ipv6_conf_table },
+ { NET_IPV6_NEIGH,  "neigh", trans_net_neigh_table },
+ { NET_IPV6_ROUTE,  "route", trans_net_ipv6_route_table },
+ { NET_IPV6_ICMP,  "icmp",  trans_net_ipv6_icmp_table },
+ { NET_IPV6_BINDV6ONLY,  "bindv6only" },
+ { NET_IPV6_IP6FRAG_HIGH_THRESH, "ip6frag_high_thresh" },
+ { NET_IPV6_IP6FRAG_LOW_THRESH, "ip6frag_low_thresh" },
+ { NET_IPV6_IP6FRAG_TIME, "ip6frag_time" },
+ { NET_IPV6_IP6FRAG_SECRET_INTERVAL, "ip6frag_secret_interval" },
+ { NET_IPV6_MLD_MAX_MSF,  "mld_max_msf" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_x25_table[] = {
+ { NET_X25_RESTART_REQUEST_TIMEOUT, "restart_request_timeout" },
+ { NET_X25_CALL_REQUEST_TIMEOUT,  "call_request_timeout" },
+ { NET_X25_RESET_REQUEST_TIMEOUT, "reset_request_timeout" },
+ { NET_X25_CLEAR_REQUEST_TIMEOUT, "clear_request_timeout" },
+ { NET_X25_ACK_HOLD_BACK_TIMEOUT, "acknowledgement_hold_back_timeout" },
+ { NET_X25_FORWARD,   "x25_forward" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_tr_table[] = {
+ { NET_TR_RIF_TIMEOUT, "rif_timeout" },
+ {}
+};
+
+
+static struct trans_ctl_table trans_net_decnet_conf_vars[] = {
+ { NET_DECNET_CONF_DEV_FORWARDING, "forwarding" },
+ { NET_DECNET_CONF_DEV_PRIORITY,  "priority" },
+ { NET_DECNET_CONF_DEV_T2,  "t2" },
+ { NET_DECNET_CONF_DEV_T3,  "t3" },
+ {}
+};
+
```

```
+static struct trans_ctl_table trans_net_decnet_conf[] = {
+ { 0, NULL, trans_net_decnet_conf_vars },
+ {}
+};
+
+static struct trans_ctl_table trans_net_decnet_table[] = {
+ { NET_DECNET_CONF,  "conf", trans_net_decnet_conf },
+ { NET_DECNET_NODE_ADDRESS, "node_address" },
+ { NET_DECNET_NODE_NAME,  "node_name" },
+ { NET_DECNET_DEFAULT_DEVICE, "default_device" },
+ { NET_DECNET_TIME_WAIT,  "time_wait" },
+ { NET_DECNET_DN_COUNT,  "dn_count" },
+ { NET_DECNET_DI_COUNT,  "di_count" },
+ { NET_DECNET_DR_COUNT,  "dr_count" },
+ { NET_DECNET_DST_GC_INTERVAL, "dst_gc_interval" },
+ { NET_DECNET_NO_FC_MAX_CWND, "no_fc_max_cwnd" },
+ { NET_DECNET_MEM,  "decnet_mem" },
+ { NET_DECNET_RMEM,  "decnet_rmem" },
+ { NET_DECNET_WMEM,  "decnet_wmem" },
+ { NET_DECNET_DEBUG_LEVEL, "debug" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_sctp_table[] = {
+ { NET_SCTP_RTO_INITIAL,  "rto_initial" },
+ { NET_SCTP_RTO_MIN,  "rto_min" },
+ { NET_SCTP_RTO_MAX,  "rto_max" },
+ { NET_SCTP_RTO_ALPHA,  "rto_alpha_exp_divisor" },
+ { NET_SCTP_RTO_BETA,  "rto_beta_exp_divisor" },
+ { NET_SCTP_VALID_COOKIE_LIFE, "valid_cookie_life" },
+ { NET_SCTP_ASSOCIATION_MAX_RETRANS, "association_max_retrans" },
+ { NET_SCTP_PATH_MAX_RETRANS, "path_max_retrans" },
+ { NET_SCTP_MAX_INIT_RETRANSMITS, "max_init_retransmits" },
+ { NET_SCTP_HB_INTERVAL,  "hb_interval" },
+ { NET_SCTP_PRESERVE_ENABLE, "cookie_preserve_enable" },
+ { NET_SCTP_MAX_BURST,  "max_burst" },
+ { NET_SCTP_ADDIP_ENABLE, "addip_enable" },
+ { NET_SCTP_PRSCTP_ENABLE, "prsctp_enable" },
+ { NET_SCTP_SNDBUF_POLICY, "sndbuf_policy" },
+ { NET_SCTP_SACK_TIMEOUT, "sack_timeout" },
+ { NET_SCTP_RCVBUF_POLICY, "rcvbuf_policy" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_llc_llc2_timeout_table[] = {
+ { NET_LLC2_ACK_TIMEOUT,  "ack" },
+ { NET_LLC2_P_TIMEOUT,  "p" },
+ { NET_LLC2_REJ_TIMEOUT,  "rej" },
```

```
+ { NET_LLC2_BUSY_TIMEOUT, "busy" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_llc_station_table[] = {
+ { NET_LLC_STATION_ACK_TIMEOUT, "ack_timeout" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_llc_llc2_table[] = {
+ { NET_LLC2,  "timeout", trans_net_llc_llc2_timeout_table },
+ {}
+};
+
+static struct trans_ctl_table trans_net_llc_table[] = {
+ { NET_LLC2,  "llc2",  trans_net_llc_llc2_table },
+ { NET_LLC_STATION, "station", trans_net_llc_station_table },
+ {}
+};
+
+static struct trans_ctl_table trans_net_netfilter_table[] = {
+ { NET_NF_CONNTRACK_MAX,    "nf_conntrack_max" },
+ { NET_NF_CONNTRACK_TCP_TIMEOUT_SYN_SENT, "nf_conntrack_tcp_timeout_syn_sent"
},
+ { NET_NF_CONNTRACK_TCP_TIMEOUT_SYN_RECV, "nf_conntrack_tcp_timeout_syn_recv"
},
+ {
NET_NF_CONNTRACK_TCP_TIMEOUT_ESTABLISHED, "nf_conntrack_tcp_timeout_establishe
d" },
+ { NET_NF_CONNTRACK_TCP_TIMEOUT_FIN_WAIT, "nf_conntrack_tcp_timeout_fin_wait" },
+ {
NET_NF_CONNTRACK_TCP_TIMEOUT_CLOSE_WAIT, "nf_conntrack_tcp_timeout_close_wait"
},
+ { NET_NF_CONNTRACK_TCP_TIMEOUT_LAST_ACK, "nf_conntrack_tcp_timeout_last_ack" },
+ { NET_NF_CONNTRACK_TCP_TIMEOUT_TIME_WAIT, "nf_conntrack_tcp_timeout_time_wait"
},
+ { NET_NF_CONNTRACK_TCP_TIMEOUT_CLOSE,  "nf_conntrack_tcp_timeout_close" },
+ { NET_NF_CONNTRACK_UDP_TIMEOUT,   "nf_conntrack_udp_timeout" },
+ { NET_NF_CONNTRACK_UDP_TIMEOUT_STREAM,  "nf_conntrack_udp_timeout_stream" },
+ { NET_NF_CONNTRACK_ICMP_TIMEOUT, "nf_conntrack_icmp_timeout" },
+ { NET_NF_CONNTRACK_GENERIC_TIMEOUT,  "nf_conntrack_generic_timeout" },
+ { NET_NF_CONNTRACK_BUCKETS,   "nf_conntrack_buckets" },
+ { NET_NF_CONNTRACK_LOG_INVALID,   "nf_conntrack_log_invalid" },
+ {
NET_NF_CONNTRACK_TCP_TIMEOUT_MAX_RETRANS, "nf_conntrack_tcp_timeout_max_retr
ans" },
+ { NET_NF_CONNTRACK_TCP_LOOSE,   "nf_conntrack_tcp_loose" },
+ { NET_NF_CONNTRACK_TCP_BE_LIBERAL,  "nf_conntrack_tcp_be_liberal" },
```

```
+ { NET_NF_CONNTRACK_TCP_MAX_RETRANS,  "nf_conntrack_tcp_max_retrans" },
+ { NET_NF_CONNTRACK_SCTP_TIMEOUT_CLOSED,  "nf_conntrack_sctp_timeout_closed" },
+ {
NET_NF_CONNTRACK_SCTP_TIMEOUT_COOKIE_WAIT, "nf_conntrack_sctp_timeout_cookie_
wait" },
+ {
NET_NF_CONNTRACK_SCTP_TIMEOUT_COOKIE_ECHOED, "nf_conntrack_sctp_timeout_coo
kie_echoed" },
+ {
NET_NF_CONNTRACK_SCTP_TIMEOUT_ESTABLISHED, "nf_conntrack_sctp_timeout_establis
hed" },
+ {
NET_NF_CONNTRACK_SCTP_TIMEOUT_SHUTDOWN_SENT, "nf_conntrack_sctp_timeout_sh
utdown_sent" },
+ {
NET_NF_CONNTRACK_SCTP_TIMEOUT_SHUTDOWN_RECD, "nf_conntrack_sctp_timeout_sh
utdown_recd" },
+ {
NET_NF_CONNTRACK_SCTP_TIMEOUT_SHUTDOWN_ACK_SENT, "nf_conntrack_sctp_timeo
ut_shutdown_ack_sent" },
+ { NET_NF_CONNTRACK_COUNT,   "nf_conntrack_count" },
+ { NET_NF_CONNTRACK_ICMPV6_TIMEOUT, "nf_conntrack_icmpv6_timeout" },
+ { NET_NF_CONNTRACK_FRAG6_TIMEOUT,  "nf_conntrack_frag6_timeout" },
+ { NET_NF_CONNTRACK_FRAG6_LOW_THRESH,  "nf_conntrack_frag6_low_thresh" },
+ { NET_NF_CONNTRACK_FRAG6_HIGH_THRESH,  "nf_conntrack_frag6_high_thresh" },
+ { NET_NF_CONNTRACK_CHECKSUM,   "nf_conntrack_checksum" },
+
+ {}
+};
+
+static struct trans_ctl_table trans_net_dccp_table[] = {
+ { NET_DCCP_DEFAULT, "default" },
+ {}
+};
+
+static struct trans_ctl_table trans_net_table[] = {
+ { NET_CORE,  "core",  trans_net_core_table },
+ /* NET_ETHER not used */
+ /* NET_802 not used */
+ { NET_UNIX,  "unix",  trans_net_unix_table },
+ { NET_IPV4,  "ipv4",  trans_net_ipv4_table },
+ { NET_IPX,  "ipx",  trans_net_ipx_table },
+ { NET_ATALK,  "atalk", trans_net_atalk_table },
+ { NET_NETROM,  "netrom", trans_net_netrom_table },
+ { NET_AX25,  "ax25",  trans_net_ax25_table },
+ { NET_BRIDGE,  "bridge", trans_net_bridge_table },
+ { NET_ROSE,  "rose",  trans_net_rose_table },
+ { NET_IPV6,  "ipv6",  trans_net_ipv6_table },
```

```
+ { NET_X25,  "x25",  trans_net_x25_table },
+ { NET_TR,  "tr",  trans_net_tr_table },
+ { NET_DECNET,  "decnet", trans_net_decnet_table },
+ /*  NET_ECONET not used */
+ { NET_SCTP,  "sctp",  trans_net_sctp_table },
+ { NET_LLC,  "llc",  trans_net_llc_table },
+ { NET_NETFILTER, "netfilter", trans_net_netfilter_table },
+ { NET_DCCP,  "dccp",  trans_net_dccp_table },
+ {}
+};
+
+static struct trans_ctl_table trans_fs_quota_table[] = {
+ { FS_DQ_LOOKUPS, "lookups" },
+ { FS_DQ_DROPS,  "drops" },
+ { FS_DQ_READS,  "reads" },
+ { FS_DQ_WRITES,  "writes" },
+ { FS_DQ_CACHE_HITS, "cache_hits" },
+ { FS_DQ_ALLOCATED, "allocated_dquots" },
+ { FS_DQ_FREE,  "free_dquots" },
+ { FS_DQ_SYNCS,  "syncs" },
+ { FS_DQ_WARNINGS, "warnings" },
+ {}
+};
+
+static struct trans_ctl_table trans_fs_xfs_table[] = {
+ { XFS_RESTRICT_CHOWN, "restrict_chown" },
+ { XFS_SGID_INHERIT, "irix_sgid_inherit" },
+ { XFS_SYMLINK_MODE, "irix_symlink_mode" },
+ { XFS_PANIC_MASK, "panic_mask" },
+
+ { XFS_ERRLEVEL,  "error_level" },
+ { XFS_SYNCD_TIMER, "xfssyncd_centisecs" },
+ { XFS_INHERIT_SYNC, "inherit_sync" },
+ { XFS_INHERIT_NODUMP, "inherit_nodump" },
+ { XFS_INHERIT_NOATIME, "inherit_noatime" },
+ { XFS_BUF_TIMER, "xfsbufd_centisecs" },
+ { XFS_BUF_AGE,  "age_buffer_centisecs" },
+ { XFS_INHERIT_NOSYM, "inherit_nosymlinks" },
+ { XFS_ROTORSTEP, "rotorstep" },
+ { XFS_INHERIT_NODFRG, "inherit_nodefrag" },
+ { XFS_FILESTREAM_TIMER, "filestream_centisecs" },
+ { XFS_STATS_CLEAR, "stats_clear" },
+ {}
+};
+
+static struct trans_ctl_table trans_fs_ocfs2_nm_table[] = {
+ { 1, "hb_ctl_path" },
+ {}
```

```
+};
+
+static struct trans_ctl_table trans_fs_ocfs2_table[] = {
+ { 1, "nm", trans_fs_ocfs2_nm_table },
+ {}
+};
+
+static struct trans_ctl_table trans_inotify_table[] = {
+ { INOTIFY_MAX_USER_INSTANCES, "max_user_instances" },
+ { INOTIFY_MAX_USER_WATCHES, "max_user_watches" },
+ { INOTIFY_MAX_QUEUED_EVENTS, "max_queued_events" },
+ {}
+};
+
+static struct trans_ctl_table trans_fs_table[] = {
+ { FS_NRINODE,  "inode-nr" },
+ { FS_STATINODE,  "inode-state" },
+ /* FS_MAXINODE unused */
+ /* FS_NRDQUOT unused */
+ /* FS_MAXDQUOT unused */
+ { FS_NRFILE,  "file-nr" },
+ { FS_MAXFILE,  "file-max" },
+ { FS_DENTRY,  "dentry-state" },
+ /* FS_NRSUPER unused */
+ /* FS_MAXUPSER unused */
+ { FS_OVERFLOWUID, "overflowuid" },
+ { FS_OVERFLOWGID, "overflowgid" },
+ { FS_LEASES,  "leases-enable" },
+ { FS_DIR_NOTIFY, "dir-notify-enable" },
+ { FS_LEASE_TIME, "lease-break-time" },
+ { FS_DQSTATS,  "quota",  trans_fs_quota_table },
+ { FS_XFS,  "xfs",   trans_fs_xfs_table },
+ { FS_AIO_NR,  "aio-nr" },
+ { FS_AIO_MAX_NR, "aio-max-nr" },
+ { FS_INOTIFY,  "inotify",  trans_inotify_table },
+ { FS_OCFS2,  "ocfs2",  trans_fs_ocfs2_table },
+ { KERN_SETUID_DUMPABLE, "suid_dumpable" },
+ {}
+};
+
+static struct trans_ctl_table trans_debug_table[] = {
+ {}
+};
+
+static struct trans_ctl_table trans_cdrom_table[] = {
+ { DEV_CDROM_INFO,  "info" },
+ { DEV_CDROM_AUTOCLOSE,  "autoclose" },
+ { DEV_CDROM_AUTOEJECT,  "autoeject" },
```

```
+ { DEV_CDROM_DEBUG,  "debug" },
+ { DEV_CDROM_LOCK,  "lock" },
+ { DEV_CDROM_CHECK_MEDIA, "check_media" },
+ {}
+};
+
+static struct trans_ctl_table trans_ipmi_table[] = {
+ { DEV_IPMI_POWEROFF_POWERCYCLE, "poweroff_powercycle" },
+ {}
+};
+
+static struct trans_ctl_table trans_mac_hid_files[] = {
+ /* DEV_MAC_HID_KEYBOARD_SENDS_LINUX_KEYCODES unused */
+ /* DEV_MAC_HID_KEYBOARD_LOCK_KEYCODES unused */
+ { DEV_MAC_HID_MOUSE_BUTTON_EMULATION, "mouse_button_emulation" },
+ { DEV_MAC_HID_MOUSE_BUTTON2_KEYCODE, "mouse_button2_keycode" },
+ { DEV_MAC_HID_MOUSE_BUTTON3_KEYCODE, "mouse_button3_keycode" },
+ /* DEV_MAC_HID_ADB_MOUSE_SENDS_KEYCODES unused */
+ {}
+};
+
+static struct trans_ctl_table trans_raid_table[] = {
+ { DEV_RAID_SPEED_LIMIT_MIN, "speed_limit_min" },
+ { DEV_RAID_SPEED_LIMIT_MAX, "speed_limit_max" },
+ {}
+};
+
+static struct trans_ctl_table trans_scsi_table[] = {
+ { DEV_SCSI_LOGGING_LEVEL, "logging_level" },
+ {}
+};
+
+static struct trans_ctl_table trans_parport_default_table[] = {
+ { DEV_PARPORT_DEFAULT_TIMESLICE, "timeslice" },
+ { DEV_PARPORT_DEFAULT_SPINTIME,  "spintime" },
+ {}
+};
+
+static struct trans_ctl_table trans_parport_device_table[] = {
+ { DEV_PARPORT_DEVICE_TIMESLICE,  "timeslice" },
+ {}
+};
+
+static struct trans_ctl_table trans_parport_devices_table[] = {
+ { DEV_PARPORT_DEVICES_ACTIVE,  "active" },
+ { 0, NULL, trans_parport_device_table },
+ {}
+};
```

```
+
+static struct trans_ctl_table trans_parport_parport_table[] = {
+ { DEV_PARPORT_SPINTIME,  "spintime" },
+ { DEV_PARPORT_BASE_ADDR, "base-addr" },
+ { DEV_PARPORT_IRQ,  "irq" },
+ { DEV_PARPORT_DMA,  "dma" },
+ { DEV_PARPORT_MODES,  "modes" },
+ { DEV_PARPORT_DEVICES,  "devices", trans_parport_devices_table },
+ { DEV_PARPORT_AUTOPROBE, "autoprobe" },
+ { DEV_PARPORT_AUTOPROBE + 1, "autoprobe0" },
+ { DEV_PARPORT_AUTOPROBE + 2, "autoprobe1" },
+ { DEV_PARPORT_AUTOPROBE + 3, "autoprobe2" },
+ { DEV_PARPORT_AUTOPROBE + 4, "autoprobe3" },
+ {}
+};
+static struct trans_ctl_table trans_parport_table[] = {
+ { DEV_PARPORT_DEFAULT, "default", trans_parport_default_table },
+ { 0, NULL, trans_parport_parport_table },
+ {}
+};
+
+static struct trans_ctl_table trans_dev_table[] = {
+ { DEV_CDROM, "cdrom", trans_cdrom_table },
+ /* DEV_HWMON unused */
+ { DEV_PARPORT, "parport", trans_parport_table },
+ { DEV_RAID, "raid",  trans_raid_table },
+ { DEV_MAC_HID, "mac_hid", trans_mac_hid_files },
+ { DEV_SCSI, "scsi",  trans_scsi_table },
+ { DEV_IPMI, "ipmi",  trans_ipmi_table },
+ {}
+};
+
+static struct trans_ctl_table trans_bus_isa_table[] = {
+ { BUS_ISA_MEM_BASE, "membase" },
+ { BUS_ISA_PORT_BASE, "portbase" },
+ { BUS_ISA_PORT_SHIFT, "portshift" },
+ {}
+};
+
+static struct trans_ctl_table trans_bus_table[] = {
+ { CTL_BUS_ISA, "isa", trans_bus_isa_table },
+ {}
+};
+
+static struct trans_ctl_table trans_arlan_conf_table0[] = {
+ { 1, "spreadingCode" },
+ { 2, "channelNumber" },
+ { 3, "scramblingDisable" },
```

```
+ { 4, "txAttenuation" },
+ { 5, "systemId" },
+ { 6, "maxDatagramSize" },
+ { 7, "maxFrameSize" },
+ { 8, "maxRetries" },
+ { 9, "receiveMode" },
+ { 10, "priority" },
+ { 11, "rootOrRepeater" },
+ { 12, "SID" },
+ { 13, "registrationMode" },
+ { 14, "registrationFill" },
+ { 15, "localTalkAddress" },
+ { 16, "codeFormat" },
+ { 17, "numChannels" },
+ { 18, "channel1" },
+ { 19, "channel2" },
+ { 20, "channel3" },
+ { 21, "channel4" },
+ { 22, "txClear" },
+ { 23, "txRetries" },
+ { 24, "txRouting" },
+ { 25, "txScrambled" },
+ { 26, "rxParameter" },
+ { 27, "txTimeoutMs" },
+ { 28, "waitCardTimeout" },
+ { 29, "channelSet" },
+ { 30, "name" },
+ { 31, "waitTime" },
+ { 32, "lParameter" },
+ { 33, "_15" },
+ { 34, "headerSize" },
+ { 36, "tx_delay_ms" },
+ { 37, "retries" },
+ { 38, "ReTransmitPacketMaxSize" },
+ { 39, "waitReTransmitPacketMaxSize" },
+ { 40, "fastReTransCount" },
+ { 41, "driverRetransmissions" },
+ { 42, "txAckTimeoutMs" },
+ { 43, "registrationInterrupts" },
+ { 44, "hardwareType" },
+ { 45, "radioType" },
+ { 46, "writeEEPROM" },
+ { 47, "writeRadioType" },
+ { 48, "entry_exit_debug" },
+ { 49, "debug" },
+ { 50, "in_speed" },
+ { 51, "out_speed" },
+ { 52, "in_speed10" },
```

```
+ { 53, "out_speed10" },
+ { 54, "in_speed_max" },
+ { 55, "out_speed_max" },
+ { 56, "measure_rate" },
+ { 57, "pre_Command_Wait" },
+ { 58, "rx_tweak1" },
+ { 59, "rx_tweak2" },
+ { 60, "tx_queue_len" },
+
+ { 150, "arlan0-txRing" },
+ { 151, "arlan0-rxRing" },
+ { 152, "arlan0-18" },
+ { 153, "arlan0-ring" },
+ { 154, "arlan0-shm-cpy" },
+ { 155, "config0" },
+ { 156, "reset0" },
+ {}
+};
+
+static struct trans_ctl_table trans_arlan_conf_table1[] = {
+ { 1, "spreadingCode" },
+ { 2, "channelNumber" },
+ { 3, "scramblingDisable" },
+ { 4, "txAttenuation" },
+ { 5, "systemId" },
+ { 6, "maxDatagramSize" },
+ { 7, "maxFrameSize" },
+ { 8, "maxRetries" },
+ { 9, "receiveMode" },
+ { 10, "priority" },
+ { 11, "rootOrRepeater" },
+ { 12, "SID" },
+ { 13, "registrationMode" },
+ { 14, "registrationFill" },
+ { 15, "localTalkAddress" },
+ { 16, "codeFormat" },
+ { 17, "numChannels" },
+ { 18, "channel1" },
+ { 19, "channel2" },
+ { 20, "channel3" },
+ { 21, "channel4" },
+ { 22, "txClear" },
+ { 23, "txRetries" },
+ { 24, "txRouting" },
+ { 25, "txScrambled" },
+ { 26, "rxParameter" },
+ { 27, "txTimeoutMs" },
+ { 28, "waitCardTimeout" },
```

```
+ { 29, "channelSet" },
+ { 30, "name" },
+ { 31, "waitTime" },
+ { 32, "lParameter" },
+ { 33, "_15" },
+ { 34, "headerSize" },
+ { 36, "tx_delay_ms" },
+ { 37, "retries" },
+ { 38, "ReTransmitPacketMaxSize" },
+ { 39, "waitReTransmitPacketMaxSize" },
+ { 40, "fastReTransCount" },
+ { 41, "driverRetransmissions" },
+ { 42, "txAckTimeoutMs" },
+ { 43, "registrationInterrupts" },
+ { 44, "hardwareType" },
+ { 45, "radioType" },
+ { 46, "writeEEPROM" },
+ { 47, "writeRadioType" },
+ { 48, "entry_exit_debug" },
+ { 49, "debug" },
+ { 50, "in_speed" },
+ { 51, "out_speed" },
+ { 52, "in_speed10" },
+ { 53, "out_speed10" },
+ { 54, "in_speed_max" },
+ { 55, "out_speed_max" },
+ { 56, "measure_rate" },
+ { 57, "pre_Command_Wait" },
+ { 58, "rx_tweak1" },
+ { 59, "rx_tweak2" },
+ { 60, "tx_queue_len" },
+
+ { 150, "arlan1-txRing" },
+ { 151, "arlan1-rxRing" },
+ { 152, "arlan1-18" },
+ { 153, "arlan1-ring" },
+ { 154, "arlan1-shm-cpy" },
+ { 155, "config1" },
+ { 156, "reset1" },
+ {}
+};
+
+static struct trans_ctl_table trans_arlan_conf_table2[] = {
+ { 1, "spreadingCode" },
+ { 2, "channelNumber" },
+ { 3, "scramblingDisable" },
+ { 4, "txAttenuation" },
+ { 5, "systemId" },
```

```
+ { 6, "maxDatagramSize" },
+ { 7, "maxFrameSize" },
+ { 8, "maxRetries" },
+ { 9, "receiveMode" },
+ { 10, "priority" },
+ { 11, "rootOrRepeater" },
+ { 12, "SID" },
+ { 13, "registrationMode" },
+ { 14, "registrationFill" },
+ { 15, "localTalkAddress" },
+ { 16, "codeFormat" },
+ { 17, "numChannels" },
+ { 18, "channel1" },
+ { 19, "channel2" },
+ { 20, "channel3" },
+ { 21, "channel4" },
+ { 22, "txClear" },
+ { 23, "txRetries" },
+ { 24, "txRouting" },
+ { 25, "txScrambled" },
+ { 26, "rxParameter" },
+ { 27, "txTimeoutMs" },
+ { 28, "waitCardTimeout" },
+ { 29, "channelSet" },
+ { 30, "name" },
+ { 31, "waitTime" },
+ { 32, "lParameter" },
+ { 33, "_15" },
+ { 34, "headerSize" },
+ { 36, "tx_delay_ms" },
+ { 37, "retries" },
+ { 38, "ReTransmitPacketMaxSize" },
+ { 39, "waitReTransmitPacketMaxSize" },
+ { 40, "fastReTransCount" },
+ { 41, "driverRetransmissions" },
+ { 42, "txAckTimeoutMs" },
+ { 43, "registrationInterrupts" },
+ { 44, "hardwareType" },
+ { 45, "radioType" },
+ { 46, "writeEEPROM" },
+ { 47, "writeRadioType" },
+ { 48, "entry_exit_debug" },
+ { 49, "debug" },
+ { 50, "in_speed" },
+ { 51, "out_speed" },
+ { 52, "in_speed10" },
+ { 53, "out_speed10" },
+ { 54, "in_speed_max" },
```

```
+ { 55, "out_speed_max" },
+ { 56, "measure_rate" },
+ { 57, "pre_Command_Wait" },
+ { 58, "rx_tweak1" },
+ { 59, "rx_tweak2" },
+ { 60, "tx_queue_len" },
+
+ { 150, "arlan2-txRing" },
+ { 151, "arlan2-rxRing" },
+ { 152, "arlan2-18" },
+ { 153, "arlan2-ring" },
+ { 154, "arlan2-shm-cpy" },
+ { 155, "config2" },
+ { 156, "reset2" },
+ {}
+};
+
+static struct trans_ctl_table trans_arlan_conf_table3[] = {
+ { 1, "spreadingCode" },
+ { 2, "channelNumber" },
+ { 3, "scramblingDisable" },
+ { 4, "txAttenuation" },
+ { 5, "systemId" },
+ { 6, "maxDatagramSize" },
+ { 7, "maxFrameSize" },
+ { 8, "maxRetries" },
+ { 9, "receiveMode" },
+ { 10, "priority" },
+ { 11, "rootOrRepeater" },
+ { 12, "SID" },
+ { 13, "registrationMode" },
+ { 14, "registrationFill" },
+ { 15, "localTalkAddress" },
+ { 16, "codeFormat" },
+ { 17, "numChannels" },
+ { 18, "channel1" },
+ { 19, "channel2" },
+ { 20, "channel3" },
+ { 21, "channel4" },
+ { 22, "txClear" },
+ { 23, "txRetries" },
+ { 24, "txRouting" },
+ { 25, "txScrambled" },
+ { 26, "rxParameter" },
+ { 27, "txTimeoutMs" },
+ { 28, "waitCardTimeout" },
+ { 29, "channelSet" },
+ { 30, "name" },
```

```
+ { 31, "waitTime" },
+ { 32, "lParameter" },
+ { 33, "_15" },
+ { 34, "headerSize" },
+ { 36, "tx_delay_ms" },
+ { 37, "retries" },
+ { 38, "ReTransmitPacketMaxSize" },
+ { 39, "waitReTransmitPacketMaxSize" },
+ { 40, "fastReTransCount" },
+ { 41, "driverRetransmissions" },
+ { 42, "txAckTimeoutMs" },
+ { 43, "registrationInterrupts" },
+ { 44, "hardwareType" },
+ { 45, "radioType" },
+ { 46, "writeEEPROM" },
+ { 47, "writeRadioType" },
+ { 48, "entry_exit_debug" },
+ { 49, "debug" },
+ { 50, "in_speed" },
+ { 51, "out_speed" },
+ { 52, "in_speed10" },
+ { 53, "out_speed10" },
+ { 54, "in_speed_max" },
+ { 55, "out_speed_max" },
+ { 56, "measure_rate" },
+ { 57, "pre_Command_Wait" },
+ { 58, "rx_tweak1" },
+ { 59, "rx_tweak2" },
+ { 60, "tx_queue_len" },
+
+ { 150, "arlan3-txRing" },
+ { 151, "arlan3-rxRing" },
+ { 152, "arlan3-18" },
+ { 153, "arlan3-ring" },
+ { 154, "arlan3-shm-cpy" },
+ { 155, "config3" },
+ { 156, "reset3" },
+ {}
+};
+
+static struct trans_ctl_table trans_arlan_table[] = {
+ { 1,  "arlan0", trans_arlan_conf_table0 },
+ { 2,  "arlan1", trans_arlan_conf_table1 },
+ { 3,  "arlan2", trans_arlan_conf_table2 },
+ { 4,  "arlan3", trans_arlan_conf_table3 },
+ {}
+};
+
```

```
+static struct trans_ctl_table trans_appldata_table[] = {
+ { CTL_APPLDATA_TIMER,  "timer" },
+ { CTL_APPLDATA_INTERVAL, "interval" },
+ { CTL_APPLDATA_OS,  "os" },
+ { CTL_APPLDATA_NET_SUM,  "net_sum" },
+ { CTL_APPLDATA_MEM,  "mem" },
+ {}
+
+};
+
+static struct trans_ctl_table trans_s390dbf_table[] = {
+ { 5678 /* CTL_S390DBF_STOPPABLE */, "debug_stoppable" },
+ { 5679 /* CTL_S390DBF_ACTIVE */, "debug_active" },
+ {}
+};
+
+static struct trans_ctl_table trans_sunrpc_table[] = {
+ { CTL_RPCDEBUG,  "rpc_debug" },
+ { CTL_NFSDEBUG,  "nfs_debug" },
+ { CTL_NFSDDEBUG, "nfsd_debug" },
+ { CTL_NLMDEBUG,  "nlm_debug" },
+ { CTL_SLOTTABLE_UDP, "udp_slot_table_entries" },
+ { CTL_SLOTTABLE_TCP, "tcp_slot_table_entries" },
+ { CTL_MIN_RESVPORT, "min_resvport" },
+ { CTL_MAX_RESVPORT, "max_resvport" },
+ {}
+};
+
+static struct trans_ctl_table trans_pm_table[] = {
+ { 1 /* CTL_PM_SUSPEND */, "suspend" },
+ { 2 /* CTL_PM_CMODE */,  "cmode" },
+ { 3 /* CTL_PM_P0 */,  "p0" },
+ { 4 /* CTL_PM_CM */,  "cm" },
+ {}
+};
+
+static struct trans_ctl_table trans_frv_table[] = {
+ { 1, "cache-mode" },
+ { 2, "pin-cxnr" },
+ {}
+};
+
+static struct trans_ctl_table trans_root_table[] = {
+ { CTL_KERN, "kernel", trans_kern_table },
+ { CTL_VM, "vm",  trans_vm_table },
+ { CTL_NET, "net",  trans_net_table },
+ /* CTL_PROC not used */
+ { CTL_FS, "fs",  trans_fs_table },
```

```
+ { CTL_DEBUG, "debug", trans_debug_table },
+ { CTL_DEV, "dev",  trans_dev_table },
+ { CTL_BUS, "bus",  trans_bus_table },
+ { CTL_ABI, "abi" },
+ /* CTL_CPU not used */
+ { CTL_ARLAN, "arlan", trans_arlan_table },
+ { CTL_APPLDATA, "appldata", trans_appldata_table },
+ { CTL_S390DBF, "s390dbf", trans_s390dbf_table },
+ { CTL_SUNRPC, "sunrpc", trans_sunrpc_table },
+ { CTL_PM, "pm",  trans_pm_table },
+ { CTL_FRV, "frv",  trans_frv_table },
+ {}
+};
+
+
+
+
+static int sysctl_depth(struct ctl_table *table)
+{
+ struct ctl_table *tmp;
+ int depth;
+
+ depth = 0;
+ for (tmp = table; tmp->parent; tmp = tmp->parent)
+  depth++;
+
+ return depth;
+}
+
+static struct ctl_table *sysctl_parent(struct ctl_table *table, int n)
+{
+ int i;
+
+ for (i = 0; table && i < n; i++)
+  table = table->parent;
+
+ return table;
+}
+
+static struct trans_ctl_table *sysctl_binary_lookup(struct ctl_table *table)
+{
+ struct ctl_table *test;
+ struct trans_ctl_table *ref;
+ int depth, cur_depth;
+
+ depth = sysctl_depth(table);
+
+ cur_depth = depth;
```

```
+ ref = trans_root_table;
+repeat:
+ test = sysctl_parent(table, cur_depth);
+ for (; ref->ctl_name || ref->procname || ref->child; ref++) {
+  int match = 0;
+
+  if (cur_depth && !ref->child)
+   continue;
+
+  if (test->procname && ref->procname &&
+   (strcmp(test->procname, ref->procname) == 0))
+   match++;
+
+  if (test->ctl_name && ref->ctl_name &&
+   (test->ctl_name == ref->ctl_name))
+   match++;
+
+  if (!ref->ctl_name && !ref->procname)
+   match++;
+
+  if (match) {
+   if (cur_depth != 0) {
+    cur_depth--;
+    ref = ref->child;
+    goto repeat;
+   }
+   goto out;
+  }
+ }
+ ref = NULL;
+out:
+ return ref;
+}
+
+static void sysctl_print_path(struct ctl_table *table)
+{
+ struct ctl_table *tmp;
+ int depth, i;
+ depth = sysctl_depth(table);
+ if (table->procname) {
+  for (i = depth; i >= 0; i--) {
+   tmp = sysctl_parent(table, i);
+   printk("/%s", tmp->procname?tmp->procname:"");
+  }
+ }
+ printk(" ");
+ if (table->ctl_name) {
+  for (i = depth; i >= 0; i--) {
```

```
+   tmp = sysctl_parent(table, i);
+   printk(".%d", tmp->ctl_name);
+  }
+ }
+}
+
+static void sysctl_repair_table(struct ctl_table *table)
+{
+ /* Don't complain about the classic default
+  * sysctl strategy routine.  Maybe later we
+  * can get the tables fixed and complain about
+  * this.
+  */
+ if (table->ctl_name && table->procname &&
+  (table->proc_handler == proc_dointvec) &&
+  (!table->strategy)) {
+  table->strategy = sysctl_data;
+ }
+}
+
+static struct ctl_table *sysctl_check_lookup(struct ctl_table *table)
+{
+ struct ctl_table_header *head;
+ struct ctl_table *ref, *test;
+ int depth, cur_depth;
+
+ depth = sysctl_depth(table);
+
+ for (head = sysctl_head_next(NULL); head;
+      head = sysctl_head_next(head)) {
+  cur_depth = depth;
+  ref = head->ctl_table;
+repeat:
+  test = sysctl_parent(table, cur_depth);
+  for (; ref->ctl_name || ref->procname; ref++) {
+   int match = 0;
+   if (cur_depth && !ref->child)
+    continue;
+
+   if (test->procname && ref->procname &&
+       (strcmp(test->procname, ref->procname) == 0))
+    match++;
+
+   if (test->ctl_name && ref->ctl_name &&
+       (test->ctl_name == ref->ctl_name))
+    match++;
+
+   if (match) {
```

```
+    if (cur_depth != 0) {
+     cur_depth--;
+     ref = ref->child;
+     goto repeat;
+    }
+    goto out;
+   }
+  }
+ }
+ ref = NULL;
+out:
+ sysctl_head_finish(head);
+ return ref;
+}
+
+static void set_fail(const char **fail, struct ctl_table *table, const char *str)
+{
+ if (*fail) {
+  printk(KERN_ERR "sysctl table check failed: ");
+  sysctl_print_path(table);
+  printk(" %s\n", *fail);
+ }
+ *fail = str;
+}
+
+static int sysctl_check_dir(struct ctl_table *table)
+{
+ struct ctl_table *ref;
+ int error;
+
+ error = 0;
+ ref = sysctl_check_lookup(table);
+ if (ref) {
+  int match = 0;
+  if (table->procname && ref->procname &&
+      (strcmp(table->procname, ref->procname) == 0))
+   match++;
+
+  if (table->ctl_name && ref->ctl_name &&
+      (table->ctl_name == ref->ctl_name))
+   match++;
+
+  if (match != 2) {
+   printk(KERN_ERR "%s: failed: ", __func__);
+   sysctl_print_path(table);
+   printk(" ref: ");
+   sysctl_print_path(ref);
+   printk("\n");
```

```
+   error = -EINVAL;
+  }
+ }
+ return error;
+}
+
+static void sysctl_check_leaf(struct ctl_table *table, const char **fail)
+{
+ struct ctl_table *ref;
+
+ ref = sysctl_check_lookup(table);
+ if (ref && (ref != table))
+  set_fail(fail, table, "Sysctl already exists");
+}
+
+static void sysctl_check_bin_path(struct ctl_table *table, const char **fail)
+{
+ struct trans_ctl_table *ref;
+
+ ref = sysctl_binary_lookup(table);
+ if (table->ctl_name && !ref)
+  set_fail(fail, table, "Unknown sysctl binary path");
+ if (ref) {
+  if (ref->procname &&
+      (!table->procname ||
+       (strcmp(table->procname, ref->procname) != 0)))
+   set_fail(fail, table, "procname does not match binary path procname");
+
+  if (ref->ctl_name &&
+      (!table->ctl_name || table->ctl_name != ref->ctl_name))
+   set_fail(fail, table, "ctl_name does not match binary path ctl_name");
+ }
+}
+
+int sysctl_check_table(struct ctl_table *table)
+{
+ int error = 0;
+ for (; table->ctl_name || table->procname; table++) {
+  const char *fail = NULL;
+
+  sysctl_repair_table(table);
+  if (table->parent) {
+   if (table->procname && !table->parent->procname)
+    set_fail(&fail, table, "Parent without procname");
+   if (table->ctl_name && !table->parent->ctl_name)
+    set_fail(&fail, table, "Parent without ctl_name");
+  }
+  if (!table->procname)
```

```
+   set_fail(&fail, table, "No procname");
+  if (table->child) {
+   if (table->data)
+    set_fail(&fail, table, "Directory with data?");
+   if (table->maxlen)
+    set_fail(&fail, table, "Directory with maxlen?");
+   if ((table->mode & (S_IRUGO|S_IXUGO)) != table->mode)
+    set_fail(&fail, table, "Writable sysctl directory");
+   if (table->proc_handler)
+    set_fail(&fail, table, "Directory with proc_handler");
+   if (table->strategy)
+    set_fail(&fail, table, "Directory with strategy");
+   if (table->extra1)
+    set_fail(&fail, table, "Directory with extra1");
+   if (table->extra2)
+    set_fail(&fail, table, "Directory with extra2");
+   if (sysctl_check_dir(table))
+    set_fail(&fail, table, "Inconsistent directory");
+  } else {
+   if ((table->strategy == sysctl_data) ||
+       (table->strategy == sysctl_string) ||
+       (table->strategy == sysctl_intvec) ||
+       (table->strategy == sysctl_jiffies) ||
+       (table->strategy == sysctl_ms_jiffies) ||
+       (table->proc_handler == proc_dostring) ||
+       (table->proc_handler == proc_dointvec) ||
+       (table->proc_handler == proc_dointvec_bset) ||
+       (table->proc_handler == proc_dointvec_minmax) ||
+       (table->proc_handler == proc_dointvec_jiffies) ||
+       (table->proc_handler == proc_dointvec_userhz_jiffies) ||
+       (table->proc_handler == proc_dointvec_ms_jiffies) ||
+       (table->proc_handler == proc_doulongvec_minmax) ||
+       (table->proc_handler == proc_doulongvec_ms_jiffies_minmax)) {
+    if (!table->data)
+     set_fail(&fail, table, "No data");
+    if (!table->maxlen)
+     set_fail(&fail, table, "No maxlen");
+   }
+   if ((table->strategy == sysctl_intvec) ||
+       (table->proc_handler == proc_dointvec_minmax) ||
+       (table->proc_handler == proc_doulongvec_minmax) ||
+       (table->proc_handler == proc_doulongvec_ms_jiffies_minmax)) {
+    if (!table->extra1)
+     set_fail(&fail, table, "No min");
+    if (!table->extra2)
+     set_fail(&fail, table, "No max");
+   }
+   if (table->ctl_name && !table->strategy)
```

```
+    set_fail(&fail, table, "Missing strategy");
+#if 0
+   if (!table->ctl_name && table->strategy)
+    set_fail(&fail, table, "Strategy without ctl_name");
+#endif
+   if (table->procname && !table->proc_handler)
+    set_fail(&fail, table, "No proc_handler");
+#if 0
+   if (!table->procname && table->proc_handler)
+    set_fail(&fail, table, "proc_handler without procname");
+#endif
+   sysctl_check_leaf(table, &fail);
+  }
+  sysctl_check_bin_path(table, &fail);
+  if (fail) {
+   set_fail(&fail, table, NULL);
+   error = -EINVAL;
+  }
+  if (table->child)
+   error |= sysctl_check_table(table->child);
+ }
+ return error;
+}
--
1.5.1.1.181.g2de0
```