Subject: [PATCH 8/8] Re-enable msgmni automatic recomputing msgmni if set to negative
Posted by Nadia Derbey on Mon, 11 Feb 2008 14:16:54 GMT
View Forum Message <> Reply to Message

[PATCH 08/08]

This patch is the enhancement as asked for by Yasunori: if msgmni is set to
a negative value, register it back into the ipcns notifier chain.

A new interface has been added to the notification mechanism:
notifier_chain_cond_register() registers a notifier block only if not already
registered. With that new interface we avoid taking care of the states changes
in procfs.

Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

---
 include/linux/ipc_namespace.h | 1
 include/linux/notifier.h | 4 +++
 ipc/ipc_sysctl.c | 45 ++++++++++++++++++++++++++++++++----------
 ipc/ipcns_notifier.c | 9 ++++++++
 kernel/notifier.c | 38 ++++++++++++++++++++++++++++++++++++
 5 files changed, 87 insertions(+), 10 deletions(-)

Index: linux-2.6.24-mm1/ipc/ipc_sysctl.c
===
--- linux-2.6.24-mm1.orig/ipc/ipc_sysctl.c 2008-02-08 16:08:32.000000000 +0100
+++ linux-2.6.24-mm1/ipc/ipc_sysctl.c 2008-02-11 14:32:09.000000000 +0100
@@ -15,6 +15,8 @@
 #include <linux/sysctl.h>
 #include <linux/uaccess.h>
 #include <linux/ipc_namespace.h>
+#include <linux/msg.h>
+#include "util.h"

 static void *get_ipc(ctl_table *table)
 {
@@ -24,6 +26,27 @@ static void *get_ipc(ctl_table *table)
  return which;
 }

+/*
+ * Routine that is called when a tunable has successfully been changed by
+ * hand and it has a callback routine registered on the ipc namespace notifier
+ * chain: we don't want such tunables to be recomputed anymore upon memory
+ * add/remove or ipc namespace creation/removal.
+ * They can come back to a recomputable state by being set to a <0 value.

```
+ */
+static void tunable_set_callback(int val)
+{
+ if (val >= 0)
+  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
+ else {
+  /*
+   * Re-enable automatic recomputing only if not already
+   * enabled.
+   */
+  recompute_msgmni(current->nsproxy->ipc_ns);
+  cond_register_ipcns_notifier(current->nsproxy->ipc_ns);
+ }
+}
+
 #ifdef CONFIG_PROC_FS
 static int proc_ipc_dointvec(ctl_table *table, int write, struct file *filp,
  void __user *buffer, size_t *lenp, loff_t *ppos)
@@ -38,17 +61,17 @@ static int proc_ipc_dointvec(ctl_table *
 static int proc_ipc_callback_dointvec(ctl_table *table, int write,
  struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos)
 {
+ struct ctl_table ipc_table;
  size_t lenp_bef = *lenp;
  int rc;

- rc = proc_ipc_dointvec(table, write, filp, buffer, lenp, ppos);
+ memcpy(&ipc_table, table, sizeof(ipc_table));
+ ipc_table.data = get_ipc(table);
+
+ rc = proc_dointvec(&ipc_table, write, filp, buffer, lenp, ppos);

  if (write && !rc && lenp_bef == *lenp)
-  /*
-   * Tunable has successfully been changed from userland:
-   * disable its automatic recomputing.
-   */
-  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
+  tunable_set_callback(*((int *)(ipc_table.data)));

  return rc;
 }
@@ -119,12 +142,14 @@ static int sysctl_ipc_registered_data(ct
  rc = sysctl_ipc_data(table, name, nlen, oldval, oldlenp, newval,
   newlen);

- if (newval && newlen && rc > 0)
+ if (newval && newlen && rc > 0) {
```

```
   /*
-   * Tunable has successfully been changed from userland:
-   * disable its automatic recomputing.
+   * Tunable has successfully been changed from userland
    */
-  unregister_ipcns_notifier(current->nsproxy->ipc_ns);
+  int *data = get_ipc(table);
+
+  tunable_set_callback(*data);
+ }

  return rc;
 }
Index: linux-2.6.24-mm1/kernel/notifier.c
===================================================================
--- linux-2.6.24-mm1.orig/kernel/notifier.c 2008-02-07 13:40:46.000000000 +0100
+++ linux-2.6.24-mm1/kernel/notifier.c 2008-02-11 14:54:34.000000000 +0100
@@ -31,6 +31,21 @@ static int notifier_chain_register(struc
  return 0;
 }

+static int notifier_chain_cond_register(struct notifier_block **nl,
+  struct notifier_block *n)
+{
+ while ((*nl) != NULL) {
+  if ((*nl) == n)
+   return 0;
+  if (n->priority > (*nl)->priority)
+   break;
+  nl = &((*nl)->next);
+ }
+ n->next = *nl;
+ rcu_assign_pointer(*nl, n);
+ return 0;
+}
+
 static int notifier_chain_unregister(struct notifier_block **nl,
   struct notifier_block *n)
 {
@@ -205,6 +220,29 @@ int blocking_notifier_chain_register(str
 EXPORT_SYMBOL_GPL(blocking_notifier_chain_register);

 /**
+ * blocking_notifier_chain_cond_register - Cond add notifier to a blocking notifier chain
+ * @nh: Pointer to head of the blocking notifier chain
+ * @n: New entry in notifier chain
+ *
+ * Adds a notifier to a blocking notifier chain, only if not already
```

```
+ * present in the chain.
+ * Must be called in process context.
+ *
+ * Currently always returns zero.
+ */
+int blocking_notifier_chain_cond_register(struct blocking_notifier_head *nh,
+  struct notifier_block *n)
+{
+ int ret;
+
+ down_write(&nh->rwsem);
+ ret = notifier_chain_cond_register(&nh->head, n);
+ up_write(&nh->rwsem);
+ return ret;
+}
+EXPORT_SYMBOL_GPL(blocking_notifier_chain_cond_register);
+
+/**
  * blocking_notifier_chain_unregister - Remove notifier from a blocking notifier chain
  * @nh: Pointer to head of the blocking notifier chain
  * @n: Entry to remove from notifier chain
Index: linux-2.6.24-mm1/include/linux/notifier.h
===================================================================
--- linux-2.6.24-mm1.orig/include/linux/notifier.h 2008-02-07 13:40:35.000000000 +0100
+++ linux-2.6.24-mm1/include/linux/notifier.h 2008-02-11 14:55:06.000000000 +0100
@@ -121,6 +121,10 @@ extern int raw_notifier_chain_register(s
 extern int srcu_notifier_chain_register(struct srcu_notifier_head *nh,
   struct notifier_block *nb);

+extern int blocking_notifier_chain_cond_register(
+  struct blocking_notifier_head *nh,
+  struct notifier_block *nb);
+
 extern int atomic_notifier_chain_unregister(struct atomic_notifier_head *nh,
   struct notifier_block *nb);
 extern int blocking_notifier_chain_unregister(struct blocking_notifier_head *nh,
Index: linux-2.6.24-mm1/include/linux/ipc_namespace.h
===================================================================
--- linux-2.6.24-mm1.orig/include/linux/ipc_namespace.h 2008-02-08 14:35:08.000000000 +0100
+++ linux-2.6.24-mm1/include/linux/ipc_namespace.h 2008-02-11 11:19:31.000000000 +0100
@@ -52,6 +52,7 @@ extern atomic_t nr_ipc_ns;
 #define INIT_IPC_NS(ns)  .ns  = &init_ipc_ns,

 extern int register_ipcns_notifier(struct ipc_namespace *);
+extern int cond_register_ipcns_notifier(struct ipc_namespace *);
 extern int unregister_ipcns_notifier(struct ipc_namespace *);
 extern int ipcns_notify(unsigned long);
```

```
Index: linux-2.6.24-mm1/ipc/ipcns_notifier.c
===================================================================
--- linux-2.6.24-mm1.orig/ipc/ipcns_notifier.c 2008-02-08 14:36:05.000000000 +0100
+++ linux-2.6.24-mm1/ipc/ipcns_notifier.c 2008-02-11 12:24:39.000000000 +0100
@@ -61,6 +61,15 @@ int register_ipcns_notifier(struct ipc_n
  return blocking_notifier_chain_register(&ipcns_chain, &ns->ipcns_nb);
 }

+int cond_register_ipcns_notifier(struct ipc_namespace *ns)
+{
+ memset(&ns->ipcns_nb, 0, sizeof(ns->ipcns_nb));
+ ns->ipcns_nb.notifier_call = ipcns_callback;
+ ns->ipcns_nb.priority = IPCNS_CALLBACK_PRI;
+ return blocking_notifier_chain_cond_register(&ipcns_chain,
+       &ns->ipcns_nb);
+}
+
 int unregister_ipcns_notifier(struct ipc_namespace *ns)
 {
  return blocking_notifier_chain_unregister(&ipcns_chain,

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