
Subject: [RFC][PATCH v2 4/7] Recomputing msgmni on memory add / remove
Posted by [Nadia Derby](#) on Thu, 31 Jan 2008 13:40:22 GMT
[View Forum Message](#) <> [Reply to Message](#)

[PATCH 04/07]

This patch introduces the registration of a callback routine that recomputes msg_ctlmni upon memory add / remove.

A single notifier block is registered in the hotplug memory chain for all the ipc namespaces.

Since the ipc namespaces are not linked together, they have their own notification chain: one notifier_block is defined per ipc namespace.

Each time an ipc namespace is created (removed) it registers (unregisters) its notifier block in (from) the ipcns chain.

The callback routine registered in the memory chain invokes the ipcns notifier chain with the IPCNS_LOWMEM event.

Each callback routine registered in the ipcns namespace, in turn, recomputes msgmni for the owning namespace.

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

include/linux/ipc.h		38	+++++
include/linux/memory.h		1	
ipc/Makefile		3	+-
ipc/ipcns_notifier.c		69	+++++
ipc/msg.c		2	-
ipc/util.c		44	+++++
ipc/util.h		2	+

7 files changed, 157 insertions(+), 2 deletions(-)

Index: linux-2.6.24/include/linux/ipc.h

=====
--- linux-2.6.24.orig/include/linux/ipc.h 2008-01-31 10:03:47.000000000 +0100
+++ linux-2.6.24/include/linux/ipc.h 2008-01-31 10:49:07.000000000 +0100
@@ -81,9 +81,20 @@ struct ipc_kludge {

```
#include <linux/kref.h>
#include <linux/spinlock.h>
+#ifdef CONFIG_MEMORY_HOTPLUG
+#include <linux/notifier.h>
+#endif /* CONFIG_MEMORY_HOTPLUG */
```

```
#define IPCMNI 32768 /* <= MAX_INT limit for ipc arrays (including sysctl changes) */
```



```

+ #endif /* CONFIG_MEMORY_HOTPLUG */
+
+ #else
+ #define INIT_IPC_NS(ns)
+ static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
Index: linux-2.6.24/include/linux/memory.h
=====
--- linux-2.6.24.orig/include/linux/memory.h 2008-01-31 10:30:37.000000000 +0100
+++ linux-2.6.24/include/linux/memory.h 2008-01-31 10:52:01.000000000 +0100
@@ -59,6 +59,7 @@ struct mem_section;
+ * order in the callback chain)
+ */
+ #define SLAB_CALLBACK_PRI 1
+ #define IPC_CALLBACK_PRI 10

+ #ifndef CONFIG_MEMORY_HOTPLUG_SPARSE
+ static inline int memory_dev_init(void)
Index: linux-2.6.24/ipc/ipcns_notifier.c
=====
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.24/ipc/ipcns_notifier.c 2008-01-31 10:58:11.000000000 +0100
@@ -0,0 +1,69 @@
+ /*
+ * linux/ipc/ipcns_notifier.c
+ * Copyright (C) 2007 BULL SA. Nadia Derbey
+ *
+ * Notification mechanism for ipc namespaces:
+ * The callback routine registered in the memory chain invokes the ipcns
+ * notifier chain with the IPCNS_MEMCHANGED event.
+ * Each callback routine registered in the ipcns namespace recomputes msgmni
+ * for the owning namespace.
+ */
+
+ #include <linux/msg.h>
+ #include <linux/rcupdate.h>
+ #include <linux/notifier.h>
+
+ #include "util.h"
+
+ static BLOCKING_NOTIFIER_HEAD(ipcns_chain);
+
+ static int ipcns_callback(struct notifier_block *self,
+ unsigned long action, void *arg)
+ {
+ struct ipc_namespace *ns;

```

```

+
+ switch (action) {
+ case IPCNS_MEMCHANGED: /* amount of lowmem has changed */
+ /*
+  * It's time to recompute msgmni
+  */
+ ns = container_of(self, struct ipc_namespace, ipcns_nb);
+ /*
+  * No need to get a reference on the ns: the 1st job of
+  * free_ipc_ns() is to unregister the callback routine.
+  * blocking_notifier_chain_unregister takes the wr lock to do
+  * it.
+  * When this callback routine is called the rd lock is held by
+  * blocking_notifier_call_chain.
+  * So the ipc ns cannot be freed while we are here.
+  */
+ recompute_msgmni(ns);
+ break;
+ default:
+ break;
+ }
+
+ return NOTIFY_OK;
+}
+
+int 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_register(&ipcns_chain, &ns->ipcns_nb);
+}
+
+int unregister_ipcns_notifier(struct ipc_namespace *ns)
+{
+ return blocking_notifier_chain_unregister(&ipcns_chain,
+     &ns->ipcns_nb);
+}
+
+int ipcns_notify(unsigned long val)
+{
+ return blocking_notifier_call_chain(&ipcns_chain, val, NULL);
+}

```

Index: linux-2.6.24/ipc/Makefile

```

=====
--- linux-2.6.24.orig/ipc/Makefile 2008-01-29 16:55:04.000000000 +0100
+++ linux-2.6.24/ipc/Makefile 2008-01-31 10:59:39.000000000 +0100
@@ -3,7 +3,8 @@

```

#

```
obj-$(CONFIG_SYSVIPC_COMPAT) += compat.o
-obj-$(CONFIG_SYSVIPC) += util.o msgutil.o msg.o sem.o shm.o
+obj_mem-$(CONFIG_MEMORY_HOTPLUG) += ipcns_notifier.o
+obj-$(CONFIG_SYSVIPC) += util.o msgutil.o msg.o sem.o shm.o $(obj_mem-y)
obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysctl.o
obj_mq-$(CONFIG_COMPAT) += compat_mq.o
obj-$(CONFIG_POSIX_QUEUE) += mqueue.o msgutil.o $(obj_mq-y)
Index: linux-2.6.24/ipc/util.c
```

```
-----
--- linux-2.6.24.orig/ipc/util.c 2008-01-31 10:05:58.000000000 +0100
```

```
+++ linux-2.6.24/ipc/util.c 2008-01-31 11:04:51.000000000 +0100
```

```
@@ -33,6 +33,7 @@
```

```
#include <linux/audit.h>
```

```
#include <linux/nsproxy.h>
```

```
#include <linux/rwsem.h>
```

```
+#include <linux/memory.h>
```

```
#include <asm/unistd.h>
```

```
@@ -54,6 +55,33 @@ struct ipc_namespace init_ipc_ns = {
atomic_t nr_ipc_ns = ATOMIC_INIT(1);
```

```
+#ifdef CONFIG_MEMORY_HOTPLUG
```

```
+
```

```
+static int ipc_memory_callback(struct notifier_block *self,
```

```
+ unsigned long action, void *arg)
```

```
+{
```

```
+ switch (action) {
```

```
+ case MEM_ONLINE: /* memory successfully brought online */
```

```
+ case MEM_OFFLINE: /* or offline: it's time to recompute msgmni */
```

```
+ /*
```

```
+ * This is done by invoking the ipcns notifier chain with the
```

```
+ * IPC_MEMCHANGED event
```

```
+ */
```

```
+ ipcns_notify(IPCNS_MEMCHANGED);
```

```
+ break;
```

```
+ case MEM_GOING_ONLINE:
```

```
+ case MEM_GOING_OFFLINE:
```

```
+ case MEM_CANCEL_ONLINE:
```

```
+ case MEM_CANCEL_OFFLINE:
```

```
+ default:
```

```
+ break;
```

```
+ }
```

```
+
```

```
+ return NOTIFY_OK;
```

```

+}
+
+ #endif /* CONFIG_MEMORY_HOTPLUG */
+
+ static struct ipc_namespace *clone_ipc_ns(struct ipc_namespace *old_ns)
+ {
+     int err;
@@ -76,6 +104,8 @@ static struct ipc_namespace *clone_ipc_n
+     if (err)
+         goto err_shm;

+ register_ipcns_notifier(ns);
+
+     kref_init(&ns->kref);
+     return ns;

@@ -111,6 +141,15 @@ void free_ipc_ns(struct kref *kref)
+     struct ipc_namespace *ns;

+     ns = container_of(kref, struct ipc_namespace, kref);
+ /*
+  * Unregistering the hotplug notifier at the beginning guarantees
+  * that the ipc namespace won't be freed while we are inside the
+  * the callback routine. Since the blocking_notifier_chain_XXX
+  * routines hold a rw lock on the notifier list,
+  * unregister_ipcns_notifier() won't take the rw lock before
+  * blocking_notifier_call_chain() has released the rd lock.
+  */
+ unregister_ipcns_notifier(ns);
+     sem_exit_ns(ns);
+     msg_exit_ns(ns);
+     shm_exit_ns(ns);
@@ -123,6 +162,9 @@ void free_ipc_ns(struct kref *kref)
+     *
+     * The various system5 IPC resources (semaphores, messages and shared
+     * memory) are initialised
+     * A callback routine is registered into the memory hotplug notifier
+     * chain: since msgmni scales to lowmem this callback routine will be
+     * called upon successful memory add / remove to recompute msgmni.
+     */

+ static int __init ipc_init(void)
@@ -130,6 +172,8 @@ static int __init ipc_init(void)
+     sem_init();
+     msg_init();
+     shm_init();
+     hotplug_memory_notifier(ipc_memory_callback, IPC_CALLBACK_PRI);
+     register_ipcns_notifier(&init_ipc_ns);

```

```
return 0;
```

```
}
```

```
__initcall(ipc_init);
```

```
Index: linux-2.6.24/ipc/msg.c
```

```
=====
```

```
--- linux-2.6.24.orig/ipc/msg.c 2008-01-31 10:08:49.000000000 +0100
```

```
+++ linux-2.6.24/ipc/msg.c 2008-01-31 11:06:17.000000000 +0100
```

```
@@ -86,7 +86,7 @@ static int sysvipc_msg_proc_show(struct
```

```
 * Also take into account the number of nsproxies created so far.
```

```
 * This should be done staying within the (MSGMNI , IPCMNI/nr_ipc_ns) range.
```

```
 */
```

```
-static void recompute_msgmni(struct ipc_namespace *ns)
```

```
+void recompute_msgmni(struct ipc_namespace *ns)
```

```
{
```

```
    struct sysinfo i;
```

```
    unsigned long allowed;
```

```
Index: linux-2.6.24/ipc/util.h
```

```
=====
```

```
--- linux-2.6.24.orig/ipc/util.h 2008-01-29 16:55:04.000000000 +0100
```

```
+++ linux-2.6.24/ipc/util.h 2008-01-31 11:07:10.000000000 +0100
```

```
@@ -134,6 +134,8 @@ extern int ipcget_new(struct ipc_namespa
```

```
extern int ipcget_public(struct ipc_namespace *, struct ipc_ids *,
```

```
    struct ipc_ops *, struct ipc_params *);
```

```
+extern void recompute_msgmni(struct ipc_namespace *);
```

```
+
```

```
static inline int ipc_buildid(int id, int seq)
```

```
{
```

```
    return SEQ_MULTIPLIER * seq + id;
```

```
--
```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
