Subject: [PATCH v7 00/10] IPC: checkpoint/restore in userspace enhancements
Posted by Stanislav Kinsbursky on Thu, 18 Oct 2012 10:22:21 GMT
View Forum Message <> Reply to Message

v7:
1) Added comments in places, when behiviour is not obvious.
2) Compilation fixed for compat layer
3) test updated to use existent header files (instead of hard-coding new
defines in case of absence).
4) comment fixed in qlge driver

v6:
1) rebased on 3.7-rc1

v5:
1) Several define-dependent compile bugs fixed
2) IPC message copy test updated
3) A couple of minor fixes.
4) Qlogic driver update: rename of its internal SEM_SET define into SEM_INIT
(compile error).

v4:
1) If MSG_COPY flag is specified, then "mtype" is not a type, but message
number to copy.
2) MSG_SET_COPY logic for sys_msgctl() was removed.

v3:
1) Copy messages to user-space under spinlock was replaced by allocation of
dummy message before queue lock and then copy of desired message to the dummy
one instead of unlinking it from queue list.
I.e. the message queue copy logic was changed: messages can be retrived one by
one (instead of receiving of the whole list at once).

This patch set is aimed to provide additional functionality for all IPC
objects,
which is required for migration of these objects by user-space
checkpoint/restore utils (CRIU).

The main problem here was impossibility to set up object id. This patch set
solves the problem in two steps:
1) Makes it possible to create new object (shared memory, semaphores set or
messages queue) with ID, equal to passed key.
2) Makes it possible to change existent object key.

Another problem was to peek messages from queues without deleting them.
This was achived by introducing of new MSG_COPY flag for sys_msgrcv(). If
MSG_COPY flag is set, then msgtyp is interpreted as message number.

The following series implements...

Stanislav Kinsbursky (10):
    ipc: remove forced assignment of selected message
    ipc: "use key as id" functionality for resource get system call introduced
    ipc: segment key change helper introduced
    ipc: add new SHM_SET command for sys_shmctl() call
    ipc: add new MSG_SET command for sys_msgctl() call
    qlge driver: rename internal SEM_SET macro to SEM_INIT
    ipc: add new SEM_SET command for sys_semctl() call
    IPC: message queue receive cleanup
    IPC: message queue copy feature introduced
    test: IPC message queue copy feture test

```
 drivers/net/ethernet/qlogic/qlge/qlge.h      |    4
 drivers/net/ethernet/qlogic/qlge/qlge_main.c |   16 +-
 include/linux/msg.h                          |    5 -
 include/uapi/linux/ipc.h                     |    1
 include/uapi/linux/msg.h                     |    2
 include/uapi/linux/sem.h                     |    1
 include/uapi/linux/shm.h                     |    1
 ipc/compat.c                                 |   54 +++---
 ipc/msg.c                                    |  117 ++++++++++---
 ipc/msgutil.c                                |   38 ++++
 ipc/sem.c                                    |   15 +-
 ipc/shm.c                                    |   18 ++
 ipc/util.c                                   |   67 +++++++-
 ipc/util.h                                   |    6 +
 security/selinux/hooks.c                     |    3
 security/smack/smack_lsm.c                   |    3
 tools/testing/selftests/ipc/Makefile         |   25 +++
 tools/testing/selftests/ipc/msgque.c         |  231 ++++++++++++++++++++++++++
 18 files changed, 526 insertions(+), 81 deletions(-)
 create mode 100644 tools/testing/selftests/ipc/Makefile
 create mode 100644 tools/testing/selftests/ipc/msgque.c
```

Subject: [PATCH v7 03/10] ipc: segment key change helper introduced
Posted by Stanislav Kinsbursky on Thu, 18 Oct 2012 10:22:51 GMT
View Forum Message <> Reply to Message

This patch introduces existent segment key changing infrastructure.
New function ipc_update_key() can be used change segment key, cuid, cgid
values. It checks for that new key is not used (except IPC_PRIVATE) prior to
set it on existent.

To make this possible, added copying of this fields from user-space in
__get_compat_ipc_perm() and __get_compat_ipc64_perm() functions. Also segment
search by key and lock were splitted into different functions, because
ipc_update_key() doesn't need to lock the segment during check that new key is
not used.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 ipc/compat.c | 6 ++++++
 ipc/util.c | 49 ++---
 ipc/util.h | 2 ++
 3 files changed, 54 insertions(+), 3 deletions(-)

diff --git a/ipc/compat.c b/ipc/compat.c
index ad9518e..af30d13 100644
--- a/ipc/compat.c
+++ b/ipc/compat.c
@@ -144,6 +144,9 @@ static inline int __get_compat_ipc64_perm(struct ipc64_perm *p64,
  err = __get_user(p64->uid, &up64->uid);
  err |= __get_user(p64->gid, &up64->gid);
  err |= __get_user(p64->mode, &up64->mode);
+ err |= __get_user(p64->cuid, &up64->cuid);
+ err |= __get_user(p64->cgid, &up64->cgid);
+ err |= __get_user(p64->key, &up64->key);
  return err;
 }

@@ -155,6 +158,9 @@ static inline int __get_compat_ipc_perm(struct ipc64_perm *p,
  err = __get_user(p->uid, &up->uid);
  err |= __get_user(p->gid, &up->gid);
  err |= __get_user(p->mode, &up->mode);
+ err |= __get_user(p->cuid, &up->cuid);
+ err |= __get_user(p->cgid, &up->cgid);
+ err |= __get_user(p->key, &up->key);
  return err;
 }

diff --git a/ipc/util.c b/ipc/util.c
index 503946e..3396ab5 100644
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -173,7 +173,7 @@ void __init ipc_init_proc_interface(const char *path, const char *header,
  * @key: The key to find
  *
  * Requires ipc_ids.rw_mutex locked.
- * Returns the LOCKED pointer to the ipc structure if found or NULL
+ * Returns the UNLOCKED pointer to the ipc structure if found or NULL
  * if not.

```
  * If key is found ipc points to the owning ipc structure
  */
@@ -195,7 +195,6 @@ static struct kern_ipc_perm *ipc_findkey(struct ipc_ids *ids, key_t key)
    continue;
   }

-  ipc_lock_by_ptr(ipc);
   return ipc;
  }

@@ -203,6 +202,27 @@ static struct kern_ipc_perm *ipc_findkey(struct ipc_ids *ids, key_t key)
 }

 /**
+ * ipc_findkey_locked - find and lock a key in an ipc identifier set
+ * @ids: Identifier set
+ * @key: The key to find
+ *
+ * Requires ipc_ids.rw_mutex locked.
+ * Returns the LOCKED pointer to the ipc structure if found or NULL
+ * if not.
+ * If key is found ipc points to the owning ipc structure
+ */
+
+static struct kern_ipc_perm *ipc_findkey_locked(struct ipc_ids *ids, key_t key)
+{
+ struct kern_ipc_perm *ipc;
+
+ ipc = ipc_findkey(ids, key);
+ if (ipc)
+  ipc_lock_by_ptr(ipc);
+ return ipc;
+}
+
+/**
  * ipc_get_maxid  - get the last assigned id
  * @ids: IPC identifier set
  *
@@ -388,7 +408,7 @@ retry:
    * a new entry + read locks are not "upgradable"
    */
   down_write(&ids->rw_mutex);
- ipcp = ipc_findkey(ids, params->key);
+ ipcp = ipc_findkey_locked(ids, params->key);
   if (ipcp == NULL) {
    /* key not used */
    if (!(flg & IPC_CREAT))
@@ -755,6 +775,29 @@ int ipcget(struct ipc_namespace *ns, struct ipc_ids *ids,
```

```
 }

 /**
+ * ipc_update_key - update the key of an IPC.
+ * @in:  the permission given as input.
+ * @out: the permission of the ipc to set.
+ *
+ * Common routine called by sys_shmctl(), sys_semctl(). sys_msgctl().
+ */
+int ipc_update_key(struct ipc_ids *ids, struct ipc64_perm *in,
+      struct kern_ipc_perm *out)
+{
+
+ if (in->key && out->key != in->key) {
+  /*
+   * Check for existent segment with the same key.
+   * Note: ipc_ids.rw_mutex is taken for write already.
+   */
+  if (ipc_findkey(ids, in->key))
+   return -EEXIST;
+  out->key = in->key;
+ }
+ return 0;
+}
+
+/**
  * ipc_update_perm - update the permissions of an IPC.
  * @in:  the permission given as input.
  * @out: the permission of the ipc to set.
diff --git a/ipc/util.h b/ipc/util.h
index 3a9e558..271bded 100644
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -126,6 +126,8 @@ struct kern_ipc_perm *ipc_lock(struct ipc_ids *, int);

 void kernel_to_ipc64_perm(struct kern_ipc_perm *in, struct ipc64_perm *out);
 void ipc64_perm_to_ipc_perm(struct ipc64_perm *in, struct ipc_perm *out);
+int ipc_update_key(struct ipc_ids *ids, struct ipc64_perm *in,
+     struct kern_ipc_perm *out);
 int ipc_update_perm(struct ipc64_perm *in, struct kern_ipc_perm *out);
 struct kern_ipc_perm *ipcctl_pre_down(struct ipc_namespace *ns,
          struct ipc_ids *ids, int id, int cmd,
```

Subject: [PATCH v7 06/10] qlge driver: rename internal SEM_SET macro to SEM_INIT
Posted by Stanislav Kinsbursky on Thu, 18 Oct 2012 10:23:06 GMT

The reason for this patch is that SET_SET is desired to be a new part IPC
sys_semctl() API.
The name itself for IPC is quite natural, because all linux-specific commands
names for IPC system calls are originally created by replacing "IPC_" part by
"SEM_"("MSG_", "SHM_") part.
So, I'm hoping, that this change doesn't really matters for "QLogic qlge NIC
HBA Driver" developers, since it's just an internal define.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 drivers/net/ethernet/qlogic/qlge/qlge.h | 4 ++--
 drivers/net/ethernet/qlogic/qlge/qlge_main.c | 16 ++++++++--------
 2 files changed, 10 insertions(+), 10 deletions(-)

```
diff --git a/drivers/net/ethernet/qlogic/qlge/qlge.h b/drivers/net/ethernet/qlogic/qlge/qlge.h
index a131d7b..6f46ea5 100644
--- a/drivers/net/ethernet/qlogic/qlge/qlge.h
+++ b/drivers/net/ethernet/qlogic/qlge/qlge.h
@@ -347,10 +347,10 @@ enum {
 enum {
  /*
   * Example:
-  * reg = SEM_XGMAC0_MASK | (SEM_SET << SEM_XGMAC0_SHIFT)
+  * reg = SEM_XGMAC0_MASK | (SEM_INIT << SEM_XGMAC0_SHIFT)
   */
  SEM_CLEAR = 0,
- SEM_SET = 1,
+ SEM_INIT = 1,
  SEM_FORCE = 3,
  SEM_XGMAC0_SHIFT = 0,
  SEM_XGMAC1_SHIFT = 2,
diff --git a/drivers/net/ethernet/qlogic/qlge/qlge_main.c
b/drivers/net/ethernet/qlogic/qlge/qlge_main.c
index b262d61..cfb0f62 100644
--- a/drivers/net/ethernet/qlogic/qlge/qlge_main.c
+++ b/drivers/net/ethernet/qlogic/qlge/qlge_main.c
@@ -109,28 +109,28 @@ static int ql_sem_trylock(struct ql_adapter *qdev, u32 sem_mask)

  switch (sem_mask) {
  case SEM_XGMAC0_MASK:
-  sem_bits = SEM_SET << SEM_XGMAC0_SHIFT;
+  sem_bits = SEM_INIT << SEM_XGMAC0_SHIFT;
   break;
  case SEM_XGMAC1_MASK:
-  sem_bits = SEM_SET << SEM_XGMAC1_SHIFT;
+  sem_bits = SEM_INIT << SEM_XGMAC1_SHIFT;
   break;
```

```
  case SEM_ICB_MASK:
-  sem_bits = SEM_SET << SEM_ICB_SHIFT;
+  sem_bits = SEM_INIT << SEM_ICB_SHIFT;
   break;
  case SEM_MAC_ADDR_MASK:
-  sem_bits = SEM_SET << SEM_MAC_ADDR_SHIFT;
+  sem_bits = SEM_INIT << SEM_MAC_ADDR_SHIFT;
   break;
  case SEM_FLASH_MASK:
-  sem_bits = SEM_SET << SEM_FLASH_SHIFT;
+  sem_bits = SEM_INIT << SEM_FLASH_SHIFT;
   break;
  case SEM_PROBE_MASK:
-  sem_bits = SEM_SET << SEM_PROBE_SHIFT;
+  sem_bits = SEM_INIT << SEM_PROBE_SHIFT;
   break;
  case SEM_RT_IDX_MASK:
-  sem_bits = SEM_SET << SEM_RT_IDX_SHIFT;
+  sem_bits = SEM_INIT << SEM_RT_IDX_SHIFT;
   break;
  case SEM_PROC_REG_MASK:
-  sem_bits = SEM_SET << SEM_PROC_REG_SHIFT;
+  sem_bits = SEM_INIT << SEM_PROC_REG_SHIFT;
   break;
  default:
   netif_alert(qdev, probe, qdev->ndev, "bad Semaphore mask!.\n");
```

Subject: Re: [PATCH v7 00/10] IPC: checkpoint/restore in userspace enhancements
Posted by ebiederm on Thu, 18 Oct 2012 12:11:08 GMT
View Forum Message <> Reply to Message

Nacked-by: "Eric W. Biederman" <ebiederm@xmission.com>

You ignored all of my feedback that the first 7 messages of your patchset are unnecessary. In particular you did not attempt to focus your patchset on those operations that are most important.

Upon examination it appears also that the 8th and 9th patches of the patchset are also unnecessary. And the 10th patch is just a test of the previous patches, making the 10th patch unneceesary without the rest.

In net this entire patchset is unnecessary and a waste of your reviewers time.

Once you have IPC checkpoint and restore working there may be a point to come back and optimize things. Please don't come back with any System V

Interprocess Communication patches until you can report how much time is saved by each and every change.

Eric