
Subject: [RFC PATCH 0/5] IPC: CRIU enhancements update
Posted by [Stanislav Kinsbursky](#) on Fri, 26 Oct 2012 11:05:38 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch set is aimed to fix some errors in previously committed "[RFC PATCH v8 0/5] IPC: checkpoint/restore in userspace enhancements" patch set.

Is particular it:

- 1) wraps new sysctl in CONFIG_CHECKPOINT_RESTORE macro and updates documentation
- 2) fixes self-test to work with new sysctls.
- 3) cleanup do_msgrcv() to make it looks less ugly around MSG_COPY feature.

The following series implements...

Stanislav Kinsbursky (5):

- ipc: wrap new sysctls for CRIU inside CONFIG_CHECKPOINT_RESTORE
- ipc: remove redundant MSG_COPY check
- test: IPC message queue copy feature test update
- Documentation: update sysctl/kernel.txt
- ipc: cleanup do_msgrcv() around MSG_COPY feature

```
Documentation/sysctl/kernel.txt | 19 ++++++++
include/linux/msg.h             |  3 +
ipc/compat.c                   |  3 +
ipc/ipc_sysctl.c               |  4 ++
ipc/msg.c                      | 84 ++++++++-----
ipc/util.c                     |  2 -
tools/testing/selftests/ipc/msgque.c | 83 ++++++++-----
7 files changed, 126 insertions(+), 72 deletions(-)
```

Subject: [PATCH 1/5] ipc: wrap new sysctls for CRIU inside
CONFIG_CHECKPOINT_RESTORE
Posted by [Stanislav Kinsbursky](#) on Fri, 26 Oct 2012 11:05:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch set wraps "msg_next_id", "sem_next_id" and "shm_next_id" inside CONFIG_CHECKPOINT_RESTORE macro.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

```
ipc/ipc_sysctl.c |  4 ++++
1 files changed, 4 insertions(+), 0 deletions(-)
```

```

diff --git a/ipc/ipc_sysctl.c b/ipc/ipc_sysctl.c
index d06d77a..130dfec 100644
--- a/ipc/ipc_sysctl.c
+++ b/ipc/ipc_sysctl.c
@@ -158,7 +158,9 @@ static int proc_ipcauto_dointvec_minmax(ctl_table *table, int write,

static int zero;
static int one = 1;
#ifdef CONFIG_CHECKPOINT_RESTORE
static int int_max = INT_MAX;
#endif

static struct ctl_table ipc_kern_table[] = {
{
@@ -228,6 +230,7 @@ static struct ctl_table ipc_kern_table[] = {
.extra1 = &zero,
.extra2 = &one,
},
#ifdef CONFIG_CHECKPOINT_RESTORE
{
.procname = "sem_next_id",
.data = &init_ipc_ns.ids[IPC_SEM_IDS].next_id,
@@ -255,6 +258,7 @@ static struct ctl_table ipc_kern_table[] = {
.extra1 = &zero,
.extra2 = &int_max,
},
#endif
}
};

```

Subject: [PATCH 3/5] test: IPC message queue copy feature test update

Posted by [Stanislav Kinsbursky](#) on Fri, 26 Oct 2012 11:06:07 GMT

[View Forum Message](#) <> [Reply to Message](#)

This update fixes coding style problems (80-characters line and others).
Also, it fixes test to work with new IPC sysctls (instead of using
experimental API logic, which was thrown away and replaced by sysctls).

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

```

include/linux/msg.h          | 3 +
ipc/compat.c                 | 3 +
ipc/msg.c                    | 2 -
ipc/util.c                   | 2 -
tools/testing/selftests/ipc/msgqueue.c | 83 ++++++-----
5 files changed, 55 insertions(+), 38 deletions(-)

```

```

diff --git a/include/linux/msg.h b/include/linux/msg.h
index f38edba..391af8d 100644
--- a/include/linux/msg.h
+++ b/include/linux/msg.h
@@ -36,6 +36,7 @@ extern long do_msgsnd(int msqid, long mtype, void __user *mtext,
    size_t msgsz, int msgflg);
extern long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
    int msgflg,
-    long (*msg_fill)(void __user *, struct msg_msg *, size_t ));
+    long (*msg_fill)(void __user *, struct msg_msg *,
+    size_t));

#endif /* _LINUX_MSG_H */
diff --git a/ipc/compat.c b/ipc/compat.c
index eb3ea16..2547f29 100644
--- a/ipc/compat.c
+++ b/ipc/compat.c
@@ -365,7 +365,8 @@ long compat_sys_msgrcv(int first, int second, int msgtyp, int third,
    uptr = compat_ptr(ipck.msgp);
    msgtyp = ipck.msgtyp;
}
- return do_msgrcv(first, uptr, second, msgtyp, third, compat_do_msg_fill);
+ return do_msgrcv(first, uptr, second, msgtyp, third,
+    compat_do_msg_fill);
}
#else
long compat_sys_semctl(int semid, int semnum, int cmd, int arg)
diff --git a/ipc/msg.c b/ipc/msg.c
index f9774ff..91873df 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -771,7 +771,7 @@ static long do_msg_fill(void __user *dest, struct msg_msg *msg, size_t
    bufsz)

long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
    int msgflg,
-    long (*msg_handler)(void __user *, struct msg_msg *, size_t ))
+    long (*msg_handler)(void __user *, struct msg_msg *, size_t))
{
    struct msg_queue *msq;
    struct msg_msg *msg;
diff --git a/ipc/util.c b/ipc/util.c
index a961e46..74e1d9c 100644
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -282,7 +282,7 @@ int ipc_addid(struct ipc_ids* ids, struct kern_ipc_perm* new, int size)

    if (next_id < 0) {

```

```

    new->seq = ids->seq++;
- if(ids->seq > ids->seq_max)
+ if (ids->seq > ids->seq_max)
    ids->seq = 0;
} else {
    new->seq = ipcid_to_seqx(next_id);
diff --git a/tools/testing/selftests/ipc/msgqueue.c b/tools/testing/selftests/ipc/msgqueue.c
index e2d6a64..d664182 100644
--- a/tools/testing/selftests/ipc/msgqueue.c
+++ b/tools/testing/selftests/ipc/msgqueue.c
@@ -3,6 +3,7 @@
#include <string.h>
#include <errno.h>
#include <linux/msg.h>
+#include <fcntl.h>

#define MAX_MSG_SIZE 32

@@ -19,9 +20,9 @@ struct msg1 {
#define ANOTHER_MSG_TYPE 26538

struct msgqueue_data {
+ key_t key;
    int msq_id;
    int qbytes;
- int kern_id;
    int qnum;
    int mode;
    struct msg1 *messages;
@@ -29,41 +30,49 @@ struct msgqueue_data {

int restore_queue(struct msgqueue_data *msgqueue)
{
- struct msqid_ds ds;
- int id, i;
+ int fd, ret, id, i;
+ char buf[32];

- id = msgget(msgqueue->msq_id,
-             msgqueue->mode | IPC_CREAT | IPC_EXCL | IPC_PRESET);
- if (id == -1) {
-     printf("Failed to create queue\n");
+ fd = open("/proc/sys/kernel/msg_next_id", O_WRONLY);
+ if (fd == -1) {
+     printf("Failed to open /proc/sys/kernel/msg_next_id\n");
+     return -errno;
+ }
+ sprintf(buf, "%d", msgqueue->msq_id);

```

```

- if (id != msgqueue->msq_id) {
-   printf("Failed to preset id (%d instead of %d)\n",
-       id, msgqueue->msq_id);
-   return -EFAULT;
+ ret = write(fd, buf, strlen(buf));
+ if (ret != strlen(buf)) {
+   printf("Failed to write to /proc/sys/kernel/msg_next_id\n");
+   return -errno;
+ }

- if (msgctl(id, MSG_STAT, &ds) < 0) {
-   printf("Failed to stat queue\n");
+ id = msgget(msgqueue->key, msgqueue->mode | IPC_CREAT | IPC_EXCL);
+ if (id == -1) {
+   printf("Failed to create queue\n");
+   return -errno;
+ }

- ds.msg_perm.key = msgqueue->msq_id;
- ds.msg_qbytes = msgqueue->qbytes;
- if (msgctl(id, MSG_SET, &ds) < 0) {
-   printf("Failed to update message key\n");
-   return -errno;
+ if (id != msgqueue->msq_id) {
+   printf("Restored queue has wrong id (%d instead of %d)\n",
+       id, msgqueue->msq_id);
+   ret = -EFAULT;
+   goto destroy;
+ }

    for (i = 0; i < msgqueue->qnum; i++) {
-   if (msgsnd(msgqueue->msq_id, &msgqueue->messages[i].mtype, msgqueue->messages[i].msize,
IPC_NOWAIT) != 0) {
+   if (msgsnd(msgqueue->msq_id, &msgqueue->messages[i].mtype,
+       msgqueue->messages[i].msize, IPC_NOWAIT) != 0) {
+       printf("msgsnd failed (%m)\n");
-   return -errno;
+   ret = -errno;
+   goto destroy;
+   };
+ }
    return 0;
+
+destroy:
+ if (msgctl(id, IPC_RMID, 0))
+   printf("Failed to destroy queue: %d\n", -errno);
+ return ret;

```

```

}

int check_and_destroy_queue(struct msgque_data *msgque)
@@ -72,7 +81,8 @@ int check_and_destroy_queue(struct msgque_data *msgque)
    int cnt = 0, ret;

    while (1) {
-   ret = msgrcv(msgque->msq_id, &message.mtype, MAX_MSG_SIZE, 0, IPC_NOWAIT);
+   ret = msgrcv(msgque->msq_id, &message.mtype, MAX_MSG_SIZE,
+   0, IPC_NOWAIT);
    if (ret < 0) {
        if (errno == ENOMSG)
            break;
@@ -81,7 +91,8 @@ int check_and_destroy_queue(struct msgque_data *msgque)
        goto err;
    }
    if (ret != msgque->messages[cnt].msize) {
-   printf("Wrong message size: %d (expected %d)\n", ret, msgque->messages[cnt].msize);
+   printf("Wrong message size: %d (expected %d)\n", ret,
+   msgque->messages[cnt].msize);
        ret = -EINVAL;
        goto err;
    }
@@ -115,15 +126,17 @@ err:

int dump_queue(struct msgque_data *msgque)
{
- struct msqid_ds ds;
+ struct msqid64_ds ds;
+ int kern_id;
    int i, ret;

- for (msgque->kern_id = 0; msgque->kern_id < 256; msgque->kern_id++) {
-   ret = msgctl(msgque->kern_id, MSG_STAT, &ds);
+ for (kern_id = 0; kern_id < 256; kern_id++) {
+   ret = msgctl(kern_id, MSG_STAT, &ds);
    if (ret < 0) {
        if (errno == -EINVAL)
            continue;
-   printf("Failed to get stats for IPC queue with id %d\n", msgque->kern_id);
+   printf("Failed to get stats for IPC queue with id %d\n",
+   kern_id);
        return -errno;
    }
}

@@ -142,7 +155,8 @@ int dump_queue(struct msgque_data *msgque)
    msgque->qbytes = ds.msg_qbytes;

```

```

    for (i = 0; i < msgqueue->qnum; i++) {
- ret = msgrcv(msgqueue->msq_id, &msgqueue->messages[i].mtype, MAX_MSG_SIZE, i,
IPC_NOWAIT | MSG_COPY);
+ ret = msgrcv(msgqueue->msq_id, &msgqueue->messages[i].mtype,
+ MAX_MSG_SIZE, i, IPC_NOWAIT | MSG_COPY);
    if (ret < 0) {
        printf("Failed to copy IPC message: %m (%d)\n", errno);
        return -errno;
@@ -158,33 +172,34 @@ int fill_msgqueue(struct msgqueue_data *msgqueue)

    msgbuf.mtype = MSG_TYPE;
    memcpy(msgbuf.mtext, TEST_STRING, sizeof(TEST_STRING));
- if (msgsnd(msgqueue->msq_id, &msgbuf.mtype, sizeof(TEST_STRING), IPC_NOWAIT) != 0) {
+ if (msgsnd(msgqueue->msq_id, &msgbuf.mtype, sizeof(TEST_STRING),
+ IPC_NOWAIT) != 0) {
    printf("First message send failed (%m)\n");
    return -errno;
};

    msgbuf.mtype = ANOTHER_MSG_TYPE;
    memcpy(msgbuf.mtext, ANOTHER_TEST_STRING, sizeof(ANOTHER_TEST_STRING));
- if (msgsnd(msgqueue->msq_id, &msgbuf.mtype, sizeof(ANOTHER_TEST_STRING),
IPC_NOWAIT) != 0) {
+ if (msgsnd(msgqueue->msq_id, &msgbuf.mtype, sizeof(ANOTHER_TEST_STRING),
+ IPC_NOWAIT) != 0) {
    printf("Second message send failed (%m)\n");
    return -errno;
};
    return 0;
}

-int main (int argc, char **argv)
+int main(int argc, char **argv)
{
- key_t key;
    int msg, pid, err;
    struct msgqueue_data msgqueue;

- key = ftok(argv[0], 822155650);
- if (key == -1) {
+ msgqueue.key = ftok(argv[0], 822155650);
+ if (msgqueue.key == -1) {
    printf("Can't make key\n");
    return -errno;
}

- msgqueue.msq_id = msgget(key, IPC_CREAT | IPC_EXCL | 0666);
+ msgqueue.msq_id = msgget(msgqueue.key, IPC_CREAT | IPC_EXCL | 0666);

```

```
if (msgqueue.msqueue_id == -1) {
    printf("Can't create queue\n");
    goto err_out;
}
```

Subject: [PATCH 4/5] Documentation: update sysctl/kernel.txt
Posted by [Stanislav Kinsbursky](#) on Fri, 26 Oct 2012 11:06:12 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch adds documentation about new "msg_next_id", "sem_next_id" and "shm_next_id" sysctls.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

Documentation/sysctl/kernel.txt | 19 ++++++
1 files changed, 19 insertions(+), 0 deletions(-)

diff --git a/Documentation/sysctl/kernel.txt b/Documentation/sysctl/kernel.txt

index 2907ba6..51b953a 100644

--- a/Documentation/sysctl/kernel.txt

+++ b/Documentation/sysctl/kernel.txt

@@ -38,6 +38,7 @@ show up in /proc/sys/kernel:

- l2cr [PPC only]
- modprobe ==> Documentation/debugging-modules.txt
- modules_disabled
- + msg_next_id [sysv ipc]
- msgmax
- msgmnb
- msgmni

@@ -62,7 +63,9 @@ show up in /proc/sys/kernel:

- rtsig-max
- rtsig-nr
- sem
- + sem_next_id [sysv ipc]
- sg-big-buff [generic SCSI device (sg)]
- + shm_next_id [sysv ipc]
- shm_rmid_forced
- shmall
- shmmax [sysv ipc]

@@ -320,6 +323,22 @@ to false.

=====

+msg_next_id, sem_next_id, and shm_next_id:

+

+These three toggles allows to specify desired id for next allocated IPC

+object: message, semaphore or shared memory respectively.

+

+By default they are equal to -1, which means generic allocation logic.

+Possible values to set are in range {0..INT_MAX}.

+

+Notes:

+1) kernel doesn't guarantee, that new object will have desired id. So,

+it's up to userspace, how to handle an object with "wrong" id.

+2) Toggle with non-default value will be set back to -1 by kernel after

+successful IPC object allocation.

+

+=====

+

nmi_watchdog:

Enables/Disables the NMI watchdog on x86 systems. When the value is

Subject: [PATCH 5/5] ipc: cleanup do_msgrcv() around MSG_COPY feature

Posted by [Stanislav Kinsbursky](#) on Fri, 26 Oct 2012 11:06:17 GMT

[View Forum Message](#) <> [Reply to Message](#)

MSG_COPY feature was developed for Checkpoint/Restart In User space project and thus wrapped in CONFIG_CHECKPOINT_RESTORE macro. But code look a bit ugly. So this patch is an attempt to cleanup do_msgrcv() a bit and make it looks better.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

ipc/msg.c | 79 ++++++++++++++++++++++++++++++++++++++-----
1 files changed, 48 insertions(+), 31 deletions(-)

diff --git a/ipc/msg.c b/ipc/msg.c

index 91873df..d20ffc7 100644

--- a/ipc/msg.c

+++ b/ipc/msg.c

@@ -769,6 +769,45 @@ static long do_msg_fill(void __user *dest, struct msg_msg *msg, size_t
bufsz)

return msgsz;

}

+#ifdef CONFIG_CHECKPOINT_RESTORE

+static inline struct msg_msg *fill_copy(unsigned long copy_nr,

+ unsigned long msg_nr,

+ struct msg_msg *msg,

+ struct msg_msg *copy)

+{

+ if (copy_nr == msg_nr)

+ return copy_msg(msg, copy);

+ return NULL;

```

+}
+
+static inline struct msg_msg *prepare_copy(void __user *buf, size_t bufsz,
+      int msgflg, long *msgtyp,
+      unsigned long *copy_number)
+{
+ struct msg_msg *copy;
+
+ *copy_number = *msgtyp;
+ *msgtyp = 0;
+ /*
+  * Create dummy message to copy real message to.
+  */
+ copy = load_msg(buf, bufsz);
+ if (!IS_ERR(copy))
+   copy->m_ts = bufsz;
+ return copy;
+}
+
+static inline void free_copy(int msgflg, struct msg_msg *copy)
+{
+ if (msgflg & MSG_COPY)
+   free_msg(copy);
+}
+
+#else
+#define free_copy(msgflg, copy) do {} while (0)
+#define prepare_copy(buf, sz, msgflg, msgtyp, copy_nr) ERR_PTR(-ENOSYS)
+#define fill_copy(copy_nr, msg_nr, msg, copy) NULL
+#endif
+
+long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
+      int msgflg,
+      long (*msg_handler)(void __user *, struct msg_msg *, size_t))
@@ -777,38 +816,22 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
+ struct msg_msg *msg;
+ int mode;
+ struct ipc_namespace *ns;
-#ifdef CONFIG_CHECKPOINT_RESTORE
- struct msg_msg *copy = NULL;
- unsigned long copy_number = 0;
-#endif
+ struct msg_msg *copy;
+ unsigned long __maybe_unused copy_number;

+ if (msqid < 0 || (long) bufsz < 0)
+   return -EINVAL;
+ if (msgflg & MSG_COPY) {
-#ifdef CONFIG_CHECKPOINT_RESTORE

```

```

- copy_number = msgtyp;
- msgtyp = 0;
-
- /*
-  * Create dummy message to copy real message to.
-  */
- copy = load_msg(buf, bufsz);
+ copy = prepare_copy(buf, bufsz, msgflg, &msgtyp, &copy_number);
  if (IS_ERR(copy))
    return PTR_ERR(copy);
- copy->m_ts = bufsz;
-#else
- return -ENOSYS;
-#endif
}
mode = convert_mode(&msgtyp, msgflg);
ns = current->nsproxy->ipc_ns;

msq = msg_lock_check(ns, msqid);
if (IS_ERR(msq)) {
-#ifdef CONFIG_CHECKPOINT_RESTORE
- if (msgflg & MSG_COPY)
- free_msg(copy);
-#endif
+ free_copy(msgflg, copy);
  return PTR_ERR(msq);
}

@@ -835,13 +858,12 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
  if (mode == SEARCH_LESSEQUAL &&
      walk_msg->m_type != 1) {
    msgtyp = walk_msg->m_type - 1;
-#ifdef CONFIG_CHECKPOINT_RESTORE
  } else if (msgflg & MSG_COPY) {
-   if (copy_number == msg_counter) {
-     msg = copy_msg(walk_msg, copy);
+     msg = fill_copy(copy_number,
+       msg_counter,
+       walk_msg, copy);
+     if (msg)
+       break;
-   }
-#endif
  } else
    break;
  msg_counter++;
@@ -857,10 +879,8 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
  msg = ERR_PTR(-E2BIG);

```

```

    goto out_unlock;
}
#ifdef CONFIG_CHECKPOINT_RESTORE
    if (msgflg & MSG_COPY)
        goto out_unlock;
#endif
    list_del(&msg->m_list);
    msq->q_qnum--;
    msq->q_rtime = get_seconds();
@@ -945,10 +965,7 @@ out_unlock:
}
}
if (IS_ERR(msg)) {
#ifdef CONFIG_CHECKPOINT_RESTORE
- if (msgflg & MSG_COPY)
- free_msg(copy);
#endif
+ free_copy(msgflg, copy);
    return PTR_ERR(msg);
}

```
