Subject: [PATCH v4 9/9] test: IPC message queue migration test
Posted by Stanislav Kinsbursky on Mon, 13 Aug 2012 12:32:22 GMT
View Forum Message <> Reply to Message

This test is a part of CRIU development test suit.
---
 tools/testing/selftests/ipc/msgque.c | 151 ++++++++++++++++++++++++++++++++++
 1 files changed, 151 insertions(+), 0 deletions(-)
 create mode 100644 tools/testing/selftests/ipc/msgque.c

diff --git a/tools/testing/selftests/ipc/msgque.c b/tools/testing/selftests/ipc/msgque.c
new file mode 100644
index 0000000..c101e25
--- /dev/null
+++ b/tools/testing/selftests/ipc/msgque.c
@@ -0,0 +1,151 @@
+#define _GNU_SOURCE
+#include <sched.h>
+
+#include <stdio.h>
+#include <string.h>
+#include <stdlib.h>
+#include <unistd.h>
+#include <sys/types.h>
+#include <sys/wait.h>
+#include <sys/sem.h>
+#include <sys/ipc.h>
+#include <sys/msg.h>
+#include <signal.h>
+#include <errno.h>
+
+#include "zdtmtst.h"
+
+const char *test_doc="Tests sysv5 msg queues support by user space checkpointing";
+const char *test_author="Stanislav Kinsbursky <skinsbursky@openvz.org>";
+
+struct msg1 {
+ long mtype;
+ char mtext[20];
+};
+#define TEST_STRING "Test sysv5 msg"
+#define MSG_TYPE 1
+
+#define ANOTHER_TEST_STRING "Yet another test sysv5 msg"
+#define ANOTHER_MSG_TYPE 26538
+
+static int test_fn(int argc, char **argv)
+{

```
+ key_t key;
+ int msg, pid;
+ struct msg1 msgbuf;
+ int chret;
+
+ key = ftok(argv[0], 822155650);
+ if (key == -1) {
+  err("Can't make key");
+  exit(1);
+ }
+
+ pid = test_fork();
+ if (pid < 0) {
+  err("Can't fork");
+  exit(1);
+ }
+
+ msg = msgget(key, IPC_CREAT | IPC_EXCL | 0666);
+ if (msg == -1) {
+  msg = msgget(key, 0666);
+  if (msg == -1) {
+   err("Can't get queue");
+   goto err_kill;
+  }
+ }
+
+ if (pid == 0) {
+  /*
+   * Here is the place where test sleeps and waits for signal.
+   * This place is used for suspend/restore test.
+   */
+  test_waitsig();
+
+  if (msgrcv(msg, &msgbuf, sizeof(TEST_STRING), MSG_TYPE, IPC_NOWAIT) == -1) {
+   fail("Child: msgrcv failed (%m)");
+   return -errno;
+  }
+
+  if (strncmp(TEST_STRING, msgbuf.mtext, sizeof(TEST_STRING))) {
+   fail("Child: the source and received strings aren't equal");
+   return -errno;
+  }
+  test_msg("Child: received %s\n", msgbuf.mtext);
+
+  msgbuf.mtype = ANOTHER_MSG_TYPE;
+  memcpy(msgbuf.mtext, ANOTHER_TEST_STRING, sizeof(ANOTHER_TEST_STRING));
+  if (msgsnd(msg, &msgbuf, sizeof(ANOTHER_TEST_STRING), IPC_NOWAIT) != 0) {
+   fail("Child: msgsnd failed (%m)");
```

```
+   return -errno;
+  };
+  pass();
+  return 0;
+ } else {
+  msgbuf.mtype = MSG_TYPE;
+  memcpy(msgbuf.mtext, TEST_STRING, sizeof(TEST_STRING));
+  if (msgsnd(msg, &msgbuf, sizeof(TEST_STRING), IPC_NOWAIT) != 0) {
+   fail("Parent: msgsnd failed (%m)");
+   goto err_kill;
+  };
+
+  msgbuf.mtype = ANOTHER_MSG_TYPE;
+  memcpy(msgbuf.mtext, ANOTHER_TEST_STRING, sizeof(ANOTHER_TEST_STRING));
+  if (msgsnd(msg, &msgbuf, sizeof(ANOTHER_TEST_STRING), IPC_NOWAIT) != 0) {
+   fail("child: msgsnd (2) failed (%m)");
+   return -errno;
+  };
+
+  test_daemon();
+  test_waitsig();
+
+  kill(pid, SIGTERM);
+
+  wait(&chret);
+  chret = WEXITSTATUS(chret);
+  if (chret) {
+   fail("Parent: child exited with non-zero code %d (%s)\n",
+        chret, strerror(chret));
+   goto out;
+  }
+
+  if (msgrcv(msg, &msgbuf, sizeof(ANOTHER_TEST_STRING), ANOTHER_MSG_TYPE,
IPC_NOWAIT) == -1) {
+   fail("Parent: msgrcv failed (%m)");
+   goto err;
+  }
+
+  if (strncmp(ANOTHER_TEST_STRING, msgbuf.mtext, sizeof(ANOTHER_TEST_STRING))) {
+   fail("Parent: the source and received strings aren't equal");
+   goto err;
+  }
+  test_msg("Parent: received %s\n", msgbuf.mtext);
+
+  pass();
+ }
+
+out:
```

```
+ if (msgctl(msg, IPC_RMID, 0)) {
+  fail("Failed to destroy message queue: %d\n", -errno);
+  return -errno;
+ }
+ return chret;
+
+err_kill:
+ kill(pid, SIGKILL);
+ wait(NULL);
+err:
+ chret = -errno;
+ goto out;
+}
+
+int main(int argc, char **argv)
+{
+#ifdef NEW_IPC_NS
+ test_init_ns(argc, argv, CLONE_NEWIPC, test_fn);
+#else
+ test_init(argc, argv);
+ test_fn();
+#endif
+ return 0;
+}
```