Subject: [PATCH] cryo: test program for udp sockets
Posted by Benjamin Thery on Thu, 19 Jun 2008 16:17:53 GMT
View Forum Message <> Reply to Message

Here is the small test program I use to test checkpoint/restart of UDP sockets with cryo.
This program can be called either as a server listening on a given port on a given adresse, or a client sending message to a given address on a given port. The client sends integer and the server sums them.

Run two instances of sockudp, one server and one client.
Checkpoint the server.
Stop both instances.
Restart server. He will wait for more messages.
Run a new client. The server will continue to sum the data it receives!

Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>

```
---
 tests/Makefile  |    2
 tests/sockudp.c |  140 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 141 insertions(+), 1 deletion(-)

Index: cryodev/tests/Makefile
===================================================================
--- cryodev.orig/tests/Makefile
+++ cryodev/tests/Makefile
@@ -1,4 +1,4 @@
-TARGETS = sleep mksysvipc pause_asm pipe
+TARGETS = sleep mksysvipc pause_asm pipe sockudp

 CFLAGS = -static

Index: cryodev/tests/sockudp.c
===================================================================
--- /dev/null
+++ cryodev/tests/sockudp.c
@@ -0,0 +1,140 @@
+/*
+ * Create an UDP socket (IPv4 or IPv6) and send/receive data to/on a given
+ * address at a given port.
+ * Options:
+ *   -C   client mode: send to the given address
+ *   -S   server mode: receive on the given address
+ *   -a   address or hostname (default: 127.0.0.1)
+ *   -p   port (default: 43210)
+ *   -n   number of messages to send/receive (default: 100)
+ *   -d   delay between messages in client mode (default: 1s)
+ *        or timeout in server mode (default: 50s)
```

```
+ */
+#include <ctype.h>
+#include <libgen.h>
+#include <netdb.h>
+#include <stdio.h>
+#include <stdlib.h>
+#include <string.h>
+#include <unistd.h>
+
+static const char* procname;
+
+void usage()
+{
+ fprintf(stderr, "Usage: %s -C|-S [-a dstaddr] "
+  "[-p dstport] [-n nmsg] [-d delay]\n", procname);
+ fprintf(stderr, "Examples:\n");
+ fprintf(stderr, "  Server IPv4: %s -S -a 127.0.0.1 -p 12345 -n 100\n", procname);
+ fprintf(stderr, "  Client IPv6: %s -C -a ::1 -p 12345 -n 20\n", procname);
+}
+
+int run(int server, const char* addr, const char *port, int family, int nr_msg, int delay)
+{
+ struct addrinfo *ai, hints;
+ int sock;
+ int i, rc;
+ ssize_t count;
+ int sum = 0;
+
+ memset(&hints, 0, sizeof(hints));
+ hints.ai_socktype = SOCK_DGRAM;
+        hints.ai_family   = family;
+
+ rc = getaddrinfo(addr, port, &hints, &ai);
+ if (rc != 0) {
+  fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rc));
+  return 1;
+ }
+
+ printf("%s addr=%s port=%s family=%s(%d)\n",
+        server ? "Listen on" : "Send to",
+        addr, port,
+        ai->ai_family == AF_INET ? "AF_INET" :
+        ai->ai_family == AF_INET6 ? "AF_INET6" : "Unknown",
+        ai->ai_family);
+
+ sock = socket(ai->ai_family, ai->ai_socktype, ai->ai_protocol);
+ if (sock < 0) {
+  perror("socket");
```

```
+  freeaddrinfo(ai);
+  return 1;
+ }
+
+ if (server) {
+  if (bind(sock,
+    (struct sockaddr*)ai->ai_addr, ai->ai_addrlen)) {
+   perror("bind");
+   return 1;
+  }
+
+  for (i = 0; i < nr_msg; i++) {
+   alarm(delay);
+   count = recv(sock, &i, sizeof(i), 0);
+   if (count == -1) {
+    perror("recv");
+    return 1;
+   }
+   sum += i;
+   printf("Data: %d\tSum: %d\t(%d)\n", i, sum, getpid());
+  }
+ } else {
+  for (i = 0; i < nr_msg; i++) {
+   count = sendto(sock, &i, sizeof(i), MSG_CONFIRM,
+          ai->ai_addr, ai->ai_addrlen);
+   if (count != sizeof(i)) {
+    perror("sendto");
+    return 1;
+   }
+   printf("Data: %d \t(%d)\n", i, getpid());
+   sleep(delay);
+  }
+ }
+ freeaddrinfo(ai);
+ return 0;
+}
+
+int main(int argc, char* argv[])
+{
+ int opt;
+ char dst_addr[256] = "127.0.0.1";
+ char dst_port[10] = "43210";
+        int nr_msg = 100;
+ int family = AF_UNSPEC;
+ int delay = 1; /* sec */
+ int server = 0;
+
+ procname = basename(argv[0]);
```

```
+
+ printf("%s: pid: %d\n", procname, getpid());
+
+ while ((opt = getopt(argc, argv, "CSa:d:hn:p:")) != -1) {
+  switch (opt) {
+  case 'C':
+   server = 0;
+   break;
+  case 'S':
+   delay = 60;
+   server = 1;
+   break;
+  case 'a':
+   strncpy(dst_addr, optarg, 256);
+   break;
+  case 'd':
+   delay = atoi(optarg);
+   break;
+  case 'n':
+   nr_msg = atoi(optarg);
+   break;
+  case 'p':
+   strncpy(dst_port,optarg, 10);
+   dst_port[9] = '\0';
+   break;
+  case 'h':
+  default: /* '?' */
+   usage();
+   return 1;
+  }
+ }
+ return run(server, dst_addr, dst_port, family, nr_msg, delay);
+}

--
```

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