
Subject: [PATCH] cryo: Improve socket support: add UDP and IPv6

Posted by [Benjamin Thery](#) on Mon, 16 Jun 2008 15:38:29 GMT

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

Today, I played a bit with inet sockets and cryo. With few modifications I managed to "checkpoint-restart" programs that use UDP sockets to listen and send data.

This patch improves socket checkpointing in cryo:

- * Save/restore socket family: we can checkpoint IPv6 sockets now.

- * Save/restore socket type: we can checkpoint UDP sockets too.

Simple test programs for UDP sockets available upon request :)

Regards,
Benjamin

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

cr.c | 63 ++++++-----
1 file changed, 50 insertions(+), 13 deletions(-)

Index: cryodev/cr.c

=====

--- cryodev.orig/cr.c

+++ cryodev/cr.c

@@ -67,11 +67,15 @@ int pagesize = 0;

```
typedef struct isockinfo_t {
    int fdnum;
-   int type; /* socket type: UDP, TCP */
+   int type; /* socket type: SOCK_STREAM, SOCK_DGRAM */
+   int family; /* socket family: AF_INET, AF_INET6 */
    int mode; /* bind, listen, connect */
    int opt; /* various sockopt */
    int backlog; /* for listen(2) */
-   struct sockaddr_in locaddr; /* for bind(2) */
+   union {
+       struct sockaddr_in saddr4;
+       struct sockaddr_in6 saddr6;
+   } locaddr; /* for bind(2) */
    char tcpstate[TPI_LEN];
} isockinfo_t;
```

@@ -179,7 +183,7 @@ int getmaps(pid_t pid, memseg_t **p)

```
int getsockinfo(pid_t pid, pinfo_t *pi, int num)
{
```

```

- struct sockaddr name;
+ struct sockaddr_storage name;
  int namelen = (int)sizeof(name), ret = 0;
  int tpi_len = TPI_LEN;
  int flag, flaglen;
@@ -188,9 +192,11 @@ int getsockinfo(pid_t pid, pinfo_t *pi,
  /* get socketname and see if it is an unix or an inet socket */
  memset(&name, 0, sizeof(name));
  PT_GETSOCKNAME(pid, num, &name, &namelen);
- t_d(name.sa_family);
+ t_d(name.ss_family);

- if (name.sa_family != (sa_family_t)AF_INET) return ret;
+ if (name.ss_family != (sa_family_t)AF_INET &&
+   name.ss_family != (sa_family_t)AF_INET6)
+ return ret;

  if (! (pi->si = (isockinfo_t *)realloc(pi->si, sizeof(isockinfo_t) * (pi->ns+1)))) return -1;
  psi = &pi->si[pi->ns];
@@ -198,11 +204,25 @@ int getsockinfo(pid_t pid, pinfo_t *pi,

  //pi->si[pi->ns].fdnum = num;
  psi->fdnum = num;
+ psi->family = name.ss_family;

  psi->mode = ISOCK_UNKNOWN;
- memcpy((void *)&psi->locaddr, (void *)&name, sizeof(name));
- if (psi->locaddr.sin_addr.s_addr != 0 || psi->locaddr.sin_port != 0) psi->mode |= ISOCK_BIND;
-
+ memcpy((void *)&psi->locaddr, (void *)&name, namelen);
+
+ if ((psi->family == AF_INET &&
+   (psi->locaddr.saddr4.sin_addr.s_addr != INADDR_ANY ||
+    psi->locaddr.saddr4.sin_port != 0)) ||
+   (psi->family == AF_INET6 &&
+    (memcmp(&psi->locaddr.saddr6.sin6_addr, &in6addr_any, 16) != 0 ||
+     psi->locaddr.saddr6.sin6_port != 0)))
+ psi->mode |= ISOCK_BIND;
+
+ flag = 0;
+ flaglen = (int)sizeof(flag);
+ PT_GETSOCKOPT(pid, num, SOL_SOCKET, SO_TYPE, &flag, &flaglen);
+ if (flag)
+ psi->type = flag;
+
  flag = 0;
  flaglen = (int)sizeof(flag);
  PT_GETSOCKOPT(pid, num, SOL_SOCKET, SO_REUSEADDR, &flag, &flaglen);

```

```

@@ -866,6 +886,7 @@ int restore_sock(int fd, pid_t pid)
    int sock;
    size_t bufsz;
    isockinfo_t *isockinfo = NULL;
+ int len;

    for (;;) {
        read_item(fd, item, sizeof(item), &buf, &bufsz);
@@ -873,20 +894,36 @@ int restore_sock(int fd, pid_t pid)
        else ITEM_SET(isockinfo, isockinfo_t);
        else break; // unknown item

- t_d(sock = PT_SOCKET(pid, AF_INET, SOCK_STREAM, 0));
+ t_d(sock = PT_SOCKET(pid, isockinfo->family, isockinfo->type, 0));
    if (sock < 0) {
        ERROR("PT_SOCKET(%d, %d, %d, %d) errno=%d: %s",
- pid, AF_INET, SOCK_STREAM, 0, sock, strerror(-sock));
+ pid, isockinfo->family, isockinfo->type, 0,
+ sock, strerror(-sock));
    }
    t_d(PT_DUP2(pid, sock, isockinfo->fdnum));
    if (sock != isockinfo->fdnum) t_d(PT_CLOSE(pid, sock));

    if (isockinfo->mode & ISOCK_BIND) {
+ struct sockaddr *sa;
+ struct sockaddr_in sin;
+ struct sockaddr_in6 sin6;

- memcpy((void *)&sin, (void *)&isockinfo->locaddr, sizeof(sin));
- if (isockinfo->mode & ISOCK_CONNECT) sin.sin_port = htons(0);
- t_d(PT_BIND(pid, isockinfo->fdnum, &sin, sizeof(sin)));
+ if (isockinfo->family == AF_INET) {
+ len = sizeof(sin);
+ memcpy((void *)&sin,
+ (void *)&isockinfo->locaddr.saddr4, len);
+ if (isockinfo->mode & ISOCK_CONNECT)
+ sin.sin_port = htons(0);
+ sa = (struct sockaddr *) &sin;
+ } else {
+ len = sizeof(sin6);
+ memcpy((void *)&sin6,
+ (void *)&isockinfo->locaddr.saddr6, len);
+ if (isockinfo->mode & ISOCK_CONNECT)
+ sin.sin_port = htons(0);
+ sa = (struct sockaddr *) &sin6;
+ }
+ t_d(PT_BIND(pid, isockinfo->fdnum, sa, len));
    }

```

```
if (isockinfo->mode & ISOCK_LISTEN) {  
    t_d(PT_LISTEN(pid, isockinfo->fdnum, 64)); //FIXME: get backlog from TPI_INFO
```

--

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