Subject: [PATCH 01/11] Introduce ctx->error to improve error reporting
Posted by Oren Laadan on Mon, 07 Feb 2011 17:21:22 GMT
View Forum Message <> Reply to Message

if ctx->errno isn't already set, the:
- ctx_set_errno() saved errno in ctx->error
- ctx_ret_errno() sets ctx->error (and returns -1)

Signed-off-by: Oren Laadan <orenl@cs.columbia.edu>
---
 common.h  |    3 ++
 restart.c |   99 +++++++

```
  int pipe_out;
@@ -257,6 +259,20 @@ static inline int ckpt_cond_fail(struct ckpt_ctx *ctx, long mask)
  return (ctx->args->fail & mask);
 }

+static inline int ctx_set_errno(struct ckpt_ctx *ctx)
+{
+ if (!ctx->error)
+  ctx->error = errno;
+ return -1;
+}
+
+static inline int ctx_ret_errno(struct ckpt_ctx *ctx, int err)
+{
+ if (!ctx->error)
+  ctx->error = err;
+ return -1;
+}
+
 static void report_exit_status(int status, char *str, int debug)
 {
  char msg[64];
@@ -426,6 +442,7 @@ int process_args(struct cr_restart_args *args)
  if (args->pidns) {
   ckpt_err("This version of restart was compiled without "
         "support for --pidns.\n");
+  errno = ENOSYS;
   return -1;
  }
 #endif
@@ -434,6 +451,7 @@ int process_args(struct cr_restart_args *args)
  if (global_debug) {
   ckpt_err("This version of restart was compiled without "
         "support for --debug.\n");
+  errno = ENOSYS;
   return -1;
  }
 #endif
@@ -446,6 +464,7 @@ int process_args(struct cr_restart_args *args)
  if (args->pids) {
   ckpt_err("This version of restart was compiled without "
         "support for --pids.\n");
+  errno = ENOSYS;
   return -1;
  }
 #endif
@@ -455,6 +474,7 @@ int process_args(struct cr_restart_args *args)
      (args->pids || args->pidns || args->show_status ||
```

```
       args->copy_status || args->freezer)) {
   ckpt_err("Invalid mix of --self with multiprocess options\n");
+  errno = EINVAL;
   return -1;
  }

@@ -703,7 +723,7 @@ static int ckpt_collect_child(struct ckpt_ctx *ctx)
   status = global_child_status;
  } else if (pid < 0) {
   ckpt_perror("WEIRD: collect child task");
-  return -1;
+  return ctx_set_errno(ctx);
  }

  return ckpt_parse_status(status, mimic, verbose);
@@ -716,18 +736,18 @@ static int ckpt_remount_devpts(struct ckpt_ctx *ctx)
  /* make sure /dev/ptmx is a link else we'll just break */
  if (lstat("/dev/ptmx", &ptystat) < 0) {
   ckpt_perror("stat /dev/ptmx");
-  return -1;
+  return ctx_set_errno(ctx);
  }
  if ((ptystat.st_mode & S_IFMT) != S_IFLNK) {
-  ckpt_err("Error: /dev/ptmx must be a link to /dev/pts/ptmx\n");
-  return -1;
+  ckpt_err("[err] /dev/ptmx must be a link to /dev/pts/ptmx\n");
+  return ctx_ret_errno(ctx, ENODEV);
  }

  /* this is unlikely, but maybe we don't want to fail */
  if (umount2("/dev/pts", MNT_DETACH) < 0) {
   if (ckpt_cond_fail(ctx, CKPT_COND_MNTPTY)) {
    ckpt_perror("umount -l /dev/pts");
-   return -1;
+   return ctx_set_errno(ctx);
   }
   if (ckpt_cond_warn(ctx, CKPT_COND_MNTPTY))
    ckpt_err("[warn] failed to un-mount old /dev/pts\n");
@@ -735,7 +755,7 @@ static int ckpt_remount_devpts(struct ckpt_ctx *ctx)
  if (mount("pts", "/dev/pts", "devpts", 0,
     "ptmxmode=666,newinstance") < 0) {
   ckpt_perror("mount -t devpts -o newinstance");
-  return -1;
+  return ctx_set_errno(ctx);
  }

  return 0;
@@ -802,6 +822,7 @static int ckpt_probe_child(pid_t pid, char *str)
```

```
  ret = waitpid(pid, &status, WNOHANG);
  if (ret == pid) {
   report_exit_status(status, str, 0);
+  errno = ECHILD;
   return -1;
  } else if (ret < 0 && errno == ECHILD) {
   ckpt_err("WEIRD: %s exited without trace (%s)\n",
@@ -809,6 +830,8 @@ static int ckpt_probe_child(pid_t pid, char *str)
   return -1;
  } else if (ret != 0) {
   ckpt_err("waitpid for %s (%s)", str, strerror(errno));
+  if (ret > 0)
+   errno = ECHILD;
   return -1;
  }
  return 0;
@@ -824,14 +847,14 @@ static int ckpt_remount_proc(struct ckpt_ctx *ctx)
  if (umount2("/proc", MNT_DETACH) < 0) {
   if (ckpt_cond_fail(ctx, CKPT_COND_MNTPROC)) {
    ckpt_perror("umount -l /proc");
-   return -1;
+   return ctx_set_errno(ctx);
   }
   if (ckpt_cond_warn(ctx, CKPT_COND_MNTPROC))
    ckpt_err("[warn] failed to un-mount old /proc\n");
  }
  if (mount("proc", "/proc", "proc", 0, NULL) < 0) {
   ckpt_perror("mount -t proc");
-  return -1;
+  return ctx_set_errno(ctx);
  }

  return 0;
@@ -903,13 +926,13 @@ static int ckpt_coordinator_pidns(struct ckpt_ctx *ctx)
   */
  if (pipe(ctx->pipe_coord) < 0) {
   ckpt_perror("pipe");
-  return -1;
+  return ctx_set_errno(ctx);
  }

  stk = genstack_alloc(PTHREAD_STACK_MIN);
  if (!stk) {
   ckpt_perror("coordinator genstack_alloc");
-  return -1;
+  return ctx_set_errno(ctx);
  }
  sp = genstack_sp(stk);
```

```
@@ -937,7 +960,7 @@ static int ckpt_coordinator_pidns(struct ckpt_ctx *ctx)
   * signal handler was plugged; verify that it's still there.
   */
  if (ckpt_probe_child(coord_pid, "coordinator") < 0)
-  return -1;
+  return ctx_set_errno(ctx);

  ctx->args->copy_status = copy;

@@ -977,7 +1000,7 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)
   * signal handler was plugged; verify that it's still there.
   */
  if (ckpt_probe_child(root_pid, "root task") < 0)
-  return -1;
+  return ctx_set_errno(ctx);

  if (ctx->args->keep_frozen)
   flags |= RESTART_FROZEN;
@@ -991,7 +1014,7 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)
   ckpt_perror("restart failed");
   ckpt_verbose("Failed\n");
   ckpt_dbg("restart failed ?\n");
-  return -1;
+  return ret;
  }

  ckpt_verbose("Success\n");
@@ -1003,7 +1026,7 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)
   /* Report success/failure to the parent */
   if (write(ctx->pipe_coord[1], &ret, sizeof(ret)) < 0) {
    ckpt_perror("failed to report status");
-   return -1;
+   return ctx_set_errno(ctx);
   }

   /*
@@ -1145,12 +1168,14 @@ static int ckpt_valid_pid(struct ckpt_ctx *ctx, pid_t pid, char *which,
int i)
 {
  if (pid < 0) {
   ckpt_err("Invalid %s %d (for task#%d)\n", which, pid, i);
+  errno = EINVAL;
   return 0;
  }
  if (!ctx->args->pidns && pid == 0) {
   if (ckpt_cond_fail(ctx, CKPT_COND_PIDZERO)) {
    ckpt_err("[err] task # %d with %s zero"
```

```
      " (requires --pidns)\n", i + 1, which);
+    errno = EINVAL;
     return 0;
    } else if (ckpt_cond_warn(ctx, CKPT_COND_PIDZERO)) {
     ckpt_err("[warn] task # %d with %s zero"
@@ -1727,13 +1752,13 @@ int ckpt_fork_stub(void *data)

   /* chroot ? */
   if ((task->flags & TASK_NEWROOT) && chroot(ctx->args->root) < 0)
-   return -1;
+   return ctx_set_errno(ctx);
   /* tasks with new pid-ns need new /proc mount */
   if ((task->flags & TASK_NEWPID) && ckpt_remount_proc(ctx) < 0)
-   return -1;
+   return ctx_set_errno(ctx);
   /* remount /dev/pts ? */
   if ((task->flags & TASK_NEWPTS) && ckpt_remount_devpts(ctx) < 0)
-   return -1;
+   return ctx_set_errno(ctx);

   /*
    * In restart into a new pid namespace (--pidns), coordinator
@@ -1755,19 +1780,21 @@ int ckpt_fork_stub(void *data)
   if (!ctx->args->pidns) {
    if (prctl(PR_SET_PDEATHSIG, SIGKILL, 0, 0, 0) < 0) {
     ckpt_perror("prctl");
-    return -1;
+    return ctx_set_errno(ctx);
    }
    if (getppid() != task->real_parent) {
     ckpt_err("[%d]: parent is MIA (%d != %d)\n",
       _getpid(), getppid(), task->real_parent);
-    return -1;
+    if (errno == 0)
+     errno = ECHILD;
+    return ctx_set_errno(ctx);
    }
   }

   /* if user requested freeze at end - add ourself to cgroup */
   if (ctx->args->freezer && freezer_register(ctx, _getpid())) {
    ckpt_err("[%d]: failed add to freezer cgroup\n", _getpid());
-   return -1;
+   return ctx_set_errno(ctx);
   }

   /* root has some extra work */
@@ -1820,12 +1847,10 @@ static pid_t ckpt_fork_child(struct ckpt_ctx *ctx, struct task *child)
```

```
 #ifndef CLONE_NEWPID
   if (child->piddepth > child->creator->piddepth) {
    ckpt_err("nested pidns but CLONE_NEWPID undefined");
-   errno = -EINVAL;
-   return -1;
+   ctx_ret_errno(ctx, ENOSYS);
   } else if (child->flags & TASK_NEWPID) {
    ckpt_err("TASK_NEWPID set but CLONE_NEWPID undefined");
-   errno = -EINVAL;
-   return -1;
+   ctx_ret_errno(ctx, ENOSYS);
   }
 #else /* CLONE_NEWPID */
   if (child->piddepth > child->creator->piddepth) {
@@ -2234,6 +2259,8 @@ static int _ckpt_read(int fd, void *buf, int count)
    continue;
   if (nread == 0 && nleft == count)
    return 0;
+  if (nread == 0)
+   errno = EIO;
   if (nread <= 0)
    return -1;
   buf += nread;
@@ -2262,10 +2289,8 @@ static int ckpt_read_obj(struct ckpt_ctx *ctx,
  ret = ckpt_read(fd, h, sizeof(*h));
  if (ret < 0)
   return ret;
- if (h->len < sizeof(*h) || h->len > n) {
-  errno = EINVAL;
-  return -1;
- }
+ if (h->len < sizeof(*h) || h->len > n)
+  return ctx_ret_errno(ctx, EINVAL);
  if (h->len == sizeof(*h))
   return 0;
  return ckpt_read(fd, buf, h->len - sizeof(*h));
@@ -2279,10 +2304,8 @@ static int ckpt_read_obj_type(struct ckpt_ctx *ctx, void *buf, int n, int
type)
  if (ret < 0)
   return ret;
- if (h->type != type) {
-  errno = EINVAL;
-  return -1;
- }
+ if (h->type != type)
+  return ctx_ret_errno(ctx, EINVAL);
  return 0;
```

```
 }

@@ -2294,10 +2317,8 @@ static int ckpt_read_obj_ptr(struct ckpt_ctx *ctx, void *buf, int n, int
type)
  ret = ckpt_read_obj(ctx, &h, buf, n + sizeof(h));
  if (ret < 0)
   return ret;
- if (h.type != type) {
-  errno = EINVAL;
-  return -1;
- }
+ if (h.type != type)
+  return ctx_ret_errno(ctx, EINVAL);
  return 0;
 }

@@ -2323,10 +2344,8 @@ static int ckpt_read_header(struct ckpt_ctx *ctx)

  if (h->constants.uts_release_len > BUFSIZE / 4 ||
      h->constants.uts_version_len > BUFSIZE / 4 ||
-     h->constants.uts_machine_len > BUFSIZE / 4) {
-  errno = EINVAL;
-  return -1;
- }
+     h->constants.uts_machine_len > BUFSIZE / 4)
+  return ctx_ret_errno(ctx, EINVAL);

  ptr = (char *) h;

--
1.7.1
```

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