Subject: [RFC][PATCH 2/2] pipe: checkpoint and restart entity
Posted by Masahiko Takahashi on Mon, 29 Jan 2007 22:49:50 GMT
View Forum Message <> Reply to Message

[RFC][PATCH 2/2] pipe: checkpoint and restart entity

These functions actually checkpoint and restart a pipe's information
and buffer data.

Masahiko.

```
--- linux-2.6.19.2/fs/pipe.c.original 2007-01-10 11:10:37.000000000 -0800
+++ linux-2.6.19.2/fs/pipe.c 2007-01-22 16:14:52.000000000 -0800
@@ -17,6 +17,8 @@
 #include <linux/highmem.h>
 #include <linux/pagemap.h>

+#include <linux/cr.h>
+
 #include <asm/uaccess.h>
 #include <asm/ioctls.h>

@@ -1029,3 +1031,129 @@ static void __exit exit_pipe_fs(void)

 fs_initcall(init_pipe_fs);
 module_exit(exit_pipe_fs);
+
+
+// -- checkpoint (ckpoint, in short)
+int dump_pipe(struct pipe_inode_info *pipe, struct cr_pipe *pipeimg)
+{
+ int err, i, n = pipe->nrbufs;
+ unsigned int buf = pipe->curbuf;
+ char *addr;
+ struct cr_pipe_buffer_info tmpbuf;
+
+ for  (i = 0 ; i < n ; i++) {
+  struct pipe_buffer *b = &pipe->bufs[buf];
+
+  tmpbuf.offset = b->offset;
+  tmpbuf.len = b->len;
+  tmpbuf.flags = b->flags;
+
+  err = b->ops->pin(pipe, b);
+  if (err)
+   return err;
+  addr = b->ops->map(pipe, b, 0);
```

```
+  if (copy_to_user(&pipeimg->bufs[i].buf,
+    addr + b->offset, b->len))
+   return -EFAULT;
+  b->ops->unmap(pipe, b, addr);
+
+  if (copy_to_user(&pipeimg->bufs[i].info,
+    &tmpbuf, sizeof(tmpbuf)))
+   return -EFAULT;
+
+  buf = (buf+1) & (PIPE_BUFFERS-1);
+ }
+ if (copy_to_user(&(pipeimg->nrbufs), &n, sizeof(n)))
+  return -EFAULT;
+
+ return n;
+}
+
+int ckpoint_pipe(struct file *filp, struct cr_pipe *img)
+{
+ struct inode *inode = filp->f_dentry->d_inode;
+ int r;
+
+ mutex_lock(&inode->i_mutex);
+ r = dump_pipe(inode->i_pipe, img);
+ mutex_unlock(&inode->i_mutex);
+
+ return r;
+}
+
+// --- restore
+static inline int install_specified_fd(int fd, struct file *file)
+{
+ current->files->next_fd = fd;
+ return (dupfd(file, fd) == fd);
+}
+
+int restore_pipebuf(struct pipe_inode_info *pipe, struct cr_pipe *pipeimg)
+{
+ int n;
+ unsigned int buf = 0;
+ struct page *page;
+ char *addr;
+
+ n = pipe->nrbufs = pipeimg->nrbufs;
+ while (n-- > 0) {
+  struct cr_pipe_buffer *a = &pipeimg->bufs[buf];
+  struct pipe_buffer *b = &pipe->bufs[buf];
+
```

```
+  page = alloc_page(GFP_HIGHUSER);
+  if (unlikely(!page)) {
+   return -ENOMEM;
+  }
+  addr = kmap(page);
+  if (copy_from_user(addr+a->info.offset, a->buf, a->info.len))
+   return -EFAULT;
+  kunmap(page);
+
+  b->page = page;
+  b->ops = &anon_pipe_buf_ops;
+  b->offset = a->info.offset;
+  b->len = a->info.len;
+  b->flags = a->info.flags;
+  buf = (buf+1) & (PIPE_BUFFERS-1);
+ }
+ return 0;
+}
+
+int restore_pipe(int *fd, struct cr_pipe *img)
+{
+ struct file *fw, *fr;
+ int r;
+
+ fw = create_write_pipe();
+ if (IS_ERR(fw))
+  return PTR_ERR(fw);
+
+ fr = create_read_pipe(fw);
+ if (IS_ERR(fr)) {
+  r = PTR_ERR(fr);
+  goto end;
+ }
+ r = restore_pipebuf(fr->f_dentry->d_inode->i_pipe, img);
+ if (r)
+  goto err_fr;
+
+ r = -EINVAL;
+ if (!install_specified_fd(fd[0], fr))
+  goto err_fdr;
+
+ if (!install_specified_fd(fd[1], fw))
+  goto err_fdw;
+
+ return fr->f_dentry->d_inode->i_pipe->nrbufs;
+
+
+err_fdw:
```

```
+ put_unused_fd(fd[1]);
+err_fdr:
+ put_unused_fd(fd[0]);
+err_fr:
+ put_filp(fr);
+end:
+ free_write_pipe(fw);
+ return r;
+}
```

File Attachments

1) patch2of2.txt, downloaded 380 times