Subject: [PATCH v3 14/16] sunrpc: make rpc_pipefs be mountable multiple times
Posted by Kirill A. Shutemov on Fri, 14 Jan 2011 13:49:12 GMT
View Forum Message <> Reply to Message

To support containers, allow multiple independent instances of
rpc_pipefs. Use '-o newinstance' to create new of the filesystem.
The same semantics as with devpts.

Signed-off-by: Kirill A. Shutemov <kas@openvz.org>
---
 net/sunrpc/rpc_pipe.c | 80 ++-
 1 files changed, 79 insertions(+), 1 deletions(-)

diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index b72cb01..02416f1 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -19,6 +19,7 @@
 #include <linux/nsproxy.h>
 #include <linux/mnt_namespace.h>
 #include <linux/fs_struct.h>
+#include <linux/parser.h>

 #include <asm/ioctls.h>
 #include <linux/fs.h>
@@ -41,6 +42,49 @@ static struct kmem_cache *rpc_inode_cachep __read_mostly;

 #define RPC_UPCALL_TIMEOUT (30*HZ)

+struct rpc_mount_opts {
+ int newinstance;
+};
+
+enum {
+ Opt_newinstance,
+
+ Opt_err
+};
+
+static const match_table_t tokens = {
+ {Opt_newinstance, "newinstance"},
+
+ {Opt_err, NULL}
+};
+
+static int
+parse_mount_options(char *data, struct rpc_mount_opts *opts)
+{

```
+ char *p;
+
+ opts->newinstance = 0;
+
+ while ((p = strsep(&data, ",")) != NULL) {
+  substring_t args[MAX_OPT_ARGS];
+  int token;
+
+  if (!*p)
+   continue;
+
+  token = match_token(p, tokens, args);
+  switch (token) {
+  case Opt_newinstance:
+   opts->newinstance = 1;
+   break;
+  default:
+   return -EINVAL;
+  }
+ }
+
+ return 0;
+}
+
 static void rpc_purge_list(struct rpc_inode *rpci, struct list_head *head,
   void (*destroy_msg)(struct rpc_pipe_msg *), int err)
 {
@@ -1093,11 +1137,45 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
  return 0;
 }

+static int
+compare_rpc_mnt_sb(struct super_block *s, void *p)
+{
+ if (init_rpc_pipefs)
+  return init_rpc_pipefs->mnt_sb == s;
+ return 0;
+}
+
 static struct dentry *
 rpc_mount(struct file_system_type *fs_type,
   int flags, const char *dev_name, void *data)
 {
- return mount_single(fs_type, flags, data, rpc_fill_super);
+ int error;
+ struct rpc_mount_opts opts;
+ struct super_block *s;
+
```

```
+ error = parse_mount_options(data, &opts);
+ if (error)
+  return ERR_PTR(error);
+
+ if (opts.newinstance)
+  s = sget(fs_type, NULL, set_anon_super, NULL);
+ else
+  s = sget(fs_type, compare_rpc_mnt_sb, set_anon_super, NULL);
+
+ if (IS_ERR(s))
+  return ERR_CAST(s);
+
+ if (!s->s_root) {
+  s->s_flags = flags;
+  error = rpc_fill_super(s, data, flags & MS_SILENT ? 1 : 0);
+  if (error) {
+   deactivate_locked_super(s);
+   return ERR_PTR(error);
+  }
+  s->s_flags |= MS_ACTIVE;
+ }
+
+ return dget(s->s_root);
 }

 static void rpc_kill_sb(struct super_block *sb)
--
1.7.3.4
```