Subject: [PATCH 2/7] cgroup: fix comments Posted by Li Zefan on Mon, 18 Feb 2008 05:49:44 GMT View Forum Message <> Reply to Message

## fix:

- comments about need\_forkexit\_callback
- comments about release agent
- typo and comment style, etc.

```
Signed-off-by: Li Zefan <lizf@cn.fujitsu.com>
```

```
---
```

```
diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
index ff9055f..2ebf7af 100644
--- a/include/linux/cgroup.h
+++ b/include/linux/cgroup.h
@ @ -175,7 +175,7 @ @ struct css set {
```

```
*
```

- \* When reading/writing to a file:
- \* the cgroup to use in file->f\_dentry->d\_parent->d\_fsdata
- + \* the cgroup to use is file->f\_dentry->d\_parent->d\_fsdata
- \* the 'cftype' of the file is file->f\_dentry->d\_fsdata

```
*/
```

```
diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index 4766bb6..0c35022 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@ @ -113,9 +113,9 @ @ static int root_count;
#define dummytop (&rootnode.top_cgroup)
```

/\* This flag indicates whether tasks in the fork and exit paths should

- \* take callback\_mutex and check for fork/exit handlers to call. This
- \* avoids us having to do extra work in the fork/exit path if none of the
- \* subsystems need to be called.
- + \* check for fork/exit handlers to call. This avoids us having to do
- + \* extra work in the fork/exit path if none of the subsystems need to
- + \* be called.

\*/

static int need\_forkexit\_callback;

@ @ -507,8 +507,8 @ @ static struct css\_set \*find\_css\_set(

- \* critical pieces of code here. The exception occurs on cgroup\_exit(),
- \* when a task in a notify\_on\_release cgroup exits. Then cgroup\_mutex

\* is taken, and if the cgroup count is zero, a usermode call made - \* to /sbin/cgroup release agent with the name of the cgroup (path - \* relative to the root of cgroup file system) as the argument. + \* to the release agent with the name of the cgroup (path relative to + \* the root of caroup file system) as the argument. \* A caroup can only be deleted if both its 'count' of using tasks \* is zero, and its list of 'children' cgroups is empty. Since all @ @ -521,7 +521,7 @ @ static struct css set \*find css set( \* The need for this exception arises from the action of \* cgroup attach task(), which overwrites one tasks cgroup pointer with - \* another. It does so using cgroup\_mutexe, however there are + \* another. It does so using cgroup\_mutex, however there are \* several performance critical places that need to reference \* task->cgroup without the expense of grabbing a system global \* mutex. Therefore except as noted below, when dereferencing or, as @ @ -1192,7 +1192,7 @ @ static void get first subsys(const struct cgroup \*cgrp, \* Attach task 'tsk' to cgroup 'cgrp' \* Call holding cgroup mutex. May take task lock of - \* the task 'pid' during call. + \* the task 'tsk' during call. \*/ int cgroup\_attach\_task(struct cgroup \*cgrp, struct task\_struct \*tsk) { @ @ -1584,12 +1584,11 @ @ static int cgroup\_create\_file(struct dentry \*dentry, int mode, } /\* - \* cgroup create dir - create a directory for an object. - \* cgrp: the cgroup we create the directory for. - \* It must have a valid ->parent field - \* And we are going to fill its ->dentry field. - \* dentry: dentry of the new cgroup - \* mode: mode to set on new directory. + \* cgroup\_create\_dir - create a directory for an object. + \* @cgrp: the cgroup we create the directory for. It must have a valid ->parent field. And we are going to fill its ->dentry field. + \* + \* @dentry: dentry of the new cgroup + \* @mode: mode to set on new directory. \*/ static int cgroup\_create\_dir(struct cgroup \*cgrp, struct dentry \*dentry, int mode) @ @ -2199,12 +2198,12 @ @ static void init cgroup css(struct cgroup subsys state \*css, }

/\*

```
- * cgroup_create - create a cgroup
- * parent: cgroup that will be parent of the new cgroup.
- * name: name of the new cgroup. Will be strcpy'ed.
- * mode: mode to set on new inode
+ * cgroup_create - create a cgroup
+ * @parent: cgroup that will be parent of the new cgroup
+ * @dentry: dentry of the new caroup
+ * @mode: mode to set on new inode
- * Must be called with the mutex on the parent inode held
+ * Must be called with the mutex on the parent inode held
 */
static long cgroup_create(struct cgroup *parent, struct dentry *dentry,
@ @ -2349,13 +2348,12 @ @ static int cgroup_rmdir(struct inode *unused_dir, struct dentry
*dentry)
 parent = cgrp->parent;
 root = cgrp->root;
 sb = root -> sb;
+
 /*
- * Call pre destroy handlers of subsys
+ * Call pre_destroy handlers of subsys. Notify subsystems
+ * that rmdir() request comes.
 */
 cgroup_call_pre_destroy(cgrp);
- /*
- * Notify subsyses that rmdir() request comes.
- */
 if (cgroup has css refs(cgrp)) {
 mutex unlock(&cgroup mutex);
@ @ -2618,7 +2616,7 @ @ static struct file_operations proc_cgroupstats_operations = {
/**
 * cgroup fork - attach newly forked task to its parents cgroup.
- * @tsk: pointer to task_struct of forking parent process.
+ * @child: pointer to task struct of forking parent process.
 * Description: A task inherits its parent's cgroup at fork().
1.5.4.rc3
```

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