Subject: [PATCH 3/7] cgroup: clean up cgroup.h
Posted by Li Zefan on Mon, 18 Feb 2008 05:49:48 GMT
View Forum Message <> Reply to Message

- replace old name 'cont' with 'cgrp' (Paul Menage did this cleanup for
  cgroup.c in commit bd89aabc6761de1c35b154fe6f914a445d301510)
- remove a duplicate declaration of cgroup_path()

Signed-off-by: Li Zefan <lizf@cn.fujitsu.com>
---
 include/linux/cgroup.h | 48 ++++++++++++++++++++++++++++++------------------
 1 files changed, 23 insertions(+), 25 deletions(-)

diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
index 2ebf7af..028ba3b 100644
--- a/include/linux/cgroup.h
+++ b/include/linux/cgroup.h
@@ -186,15 +186,15 @@ struct cftype {
  char name[MAX_CFTYPE_NAME];
  int private;
  int (*open) (struct inode *inode, struct file *file);
- ssize_t (*read) (struct cgroup *cont, struct cftype *cft,
+ ssize_t (*read) (struct cgroup *cgrp, struct cftype *cft,
     struct file *file,
     char __user *buf, size_t nbytes, loff_t *ppos);
  /*
   * read_uint() is a shortcut for the common case of returning a
   * single integer. Use it in place of read()
   */
- u64 (*read_uint) (struct cgroup *cont, struct cftype *cft);
- ssize_t (*write) (struct cgroup *cont, struct cftype *cft,
+ u64 (*read_uint) (struct cgroup *cgrp, struct cftype *cft);
+ ssize_t (*write) (struct cgroup *cgrp, struct cftype *cft,
      struct file *file,
      const char __user *buf, size_t nbytes, loff_t *ppos);

@@ -203,7 +203,7 @@ struct cftype {
   * a single integer (as parsed by simple_strtoull) from
   * userspace. Use in place of write(); return 0 or error.
   */
- int (*write_uint) (struct cgroup *cont, struct cftype *cft, u64 val);
+ int (*write_uint) (struct cgroup *cgrp, struct cftype *cft, u64 val);

  int (*release) (struct inode *inode, struct file *file);
 };
@@ -218,41 +218,41 @@ struct cgroup_scanner {

 /* Add a new file to the given cgroup directory. Should only be

```
  * called by subsystems from within a populate() method */
-int cgroup_add_file(struct cgroup *cont, struct cgroup_subsys *subsys,
+int cgroup_add_file(struct cgroup *cgrp, struct cgroup_subsys *subsys,
          const struct cftype *cft);

 /* Add a set of new files to the given cgroup directory. Should
  * only be called by subsystems from within a populate() method */
-int cgroup_add_files(struct cgroup *cont,
+int cgroup_add_files(struct cgroup *cgrp,
    struct cgroup_subsys *subsys,
    const struct cftype cft[],
    int count);

-int cgroup_is_removed(const struct cgroup *cont);
+int cgroup_is_removed(const struct cgroup *cgrp);

-int cgroup_path(const struct cgroup *cont, char *buf, int buflen);
+int cgroup_path(const struct cgroup *cgrp, char *buf, int buflen);

-int cgroup_task_count(const struct cgroup *cont);
+int cgroup_task_count(const struct cgroup *cgrp);

 /* Return true if the cgroup is a descendant of the current cgroup */
-int cgroup_is_descendant(const struct cgroup *cont);
+int cgroup_is_descendant(const struct cgroup *cgrp);

 /* Control Group subsystem type. See Documentation/cgroups.txt for details */

 struct cgroup_subsys {
  struct cgroup_subsys_state *(*create)(struct cgroup_subsys *ss,
-        struct cgroup *cont);
- void (*pre_destroy)(struct cgroup_subsys *ss, struct cgroup *cont);
- void (*destroy)(struct cgroup_subsys *ss, struct cgroup *cont);
+        struct cgroup *cgrp);
+ void (*pre_destroy)(struct cgroup_subsys *ss, struct cgroup *cgrp);
+ void (*destroy)(struct cgroup_subsys *ss, struct cgroup *cgrp);
  int (*can_attach)(struct cgroup_subsys *ss,
-     struct cgroup *cont, struct task_struct *tsk);
- void (*attach)(struct cgroup_subsys *ss, struct cgroup *cont,
-    struct cgroup *old_cont, struct task_struct *tsk);
+     struct cgroup *cgrp, struct task_struct *tsk);
+ void (*attach)(struct cgroup_subsys *ss, struct cgroup *cgrp,
+    struct cgroup *old_cgrp, struct task_struct *tsk);
  void (*fork)(struct cgroup_subsys *ss, struct task_struct *task);
  void (*exit)(struct cgroup_subsys *ss, struct task_struct *task);
  int (*populate)(struct cgroup_subsys *ss,
-    struct cgroup *cont);
- void (*post_clone)(struct cgroup_subsys *ss, struct cgroup *cont);
```

```
+   struct cgroup *cgrp);
+ void (*post_clone)(struct cgroup_subsys *ss, struct cgroup *cgrp);
  void (*bind)(struct cgroup_subsys *ss, struct cgroup *root);
  int subsys_id;
  int active;
@@ -273,9 +273,9 @@ struct cgroup_subsys {
 #undef SUBSYS

 static inline struct cgroup_subsys_state *cgroup_subsys_state(
- struct cgroup *cont, int subsys_id)
+ struct cgroup *cgrp, int subsys_id)
 {
- return cont->subsys[subsys_id];
+ return cgrp->subsys[subsys_id];
 }

 static inline struct cgroup_subsys_state *task_subsys_state(
@@ -290,8 +290,6 @@ static inline struct cgroup* task_cgroup(struct task_struct *task,
  return task_subsys_state(task, subsys_id)->cgroup;
 }

-int cgroup_path(const struct cgroup *cont, char *buf, int buflen);
-
 int cgroup_clone(struct task_struct *tsk, struct cgroup_subsys *ss);

 /* A cgroup_iter should be treated as an opaque object */
@@ -313,10 +311,10 @@ struct cgroup_iter {
  *     - cgroup_scan_tasks() holds the css_set_lock when calling the test_task()
  *       callback, but not while calling the process_task() callback.
  */
-void cgroup_iter_start(struct cgroup *cont, struct cgroup_iter *it);
-struct task_struct *cgroup_iter_next(struct cgroup *cont,
+void cgroup_iter_start(struct cgroup *cgrp, struct cgroup_iter *it);
+struct task_struct *cgroup_iter_next(struct cgroup *cgrp,
      struct cgroup_iter *it);
-void cgroup_iter_end(struct cgroup *cont, struct cgroup_iter *it);
+void cgroup_iter_end(struct cgroup *cgrp, struct cgroup_iter *it);
 int cgroup_scan_tasks(struct cgroup_scanner *scan);
 int cgroup_attach_task(struct cgroup *, struct task_struct *);

--
1.5.4.rc3
```