Subject: [PATCH 00/10] CGroup API files: Various cleanup to CGroup control files
Posted by Paul Menage on Sat, 23 Feb 2008 22:47:25 GMT
View Forum Message <> Reply to Message

This patchset is a roll-up of the non-contraversial items of the
various patches that I've sent out recently, fixed according to the
feedback received.

In summary they are:

- general rename of read_uint/write_uint to read_u64/write_u64

- use these methods for cpusets and memory controller files

- add a read_map cgroup file method, and use it in the memory
  controller

- move the "releasable" control file to the debug subsystem

- make the debug subsystem config option default to "n"

The only user-visible changes are the movement of the "releasable"
file and the fact that some write_u64()-based control files are now
more forgiving of additional whitespace at the end of their input.

Signed-off-by: Paul Menage <menage@google.com>

--

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

Subject: [PATCH 02/10] CGroup API files: Add res_counter_read_u64()
Posted by Paul Menage on Sat, 23 Feb 2008 22:47:27 GMT
View Forum Message <> Reply to Message

Adds a function for returning the value of a resource counter member,
in a form suitable for use in a cgroup read_u64 control file method.

Signed-off-by: Paul Menage <menage@google.com>

---
 include/linux/res_counter.h | 5 ++++-
 kernel/res_counter.c | 5 +++++
 2 files changed, 9 insertions(+), 1 deletion(-)

```
Index: cgroup-2.6.25-rc2-mm1/include/linux/res_counter.h
===================================================================
--- cgroup-2.6.25-rc2-mm1.orig/include/linux/res_counter.h
+++ cgroup-2.6.25-rc2-mm1/include/linux/res_counter.h
@@ -39,8 +39,9 @@ struct res_counter {
  spinlock_t lock;
 };

-/*
+/**
  * Helpers to interact with userspace
+ * res_counter_read_u64() - returns the value of the specified member.
  * res_counter_read/_write - put/get the specified fields from the
  * res_counter struct to/from the user
  *
@@ -51,6 +52,8 @@ struct res_counter {
  * @pos:     and the offset.
  */

+u64 res_counter_read_u64(struct res_counter *counter, int member);
+
 ssize_t res_counter_read(struct res_counter *counter, int member,
   const char __user *buf, size_t nbytes, loff_t *pos,
   int (*read_strategy)(unsigned long long val, char *s));
Index: cgroup-2.6.25-rc2-mm1/kernel/res_counter.c
===================================================================
--- cgroup-2.6.25-rc2-mm1.orig/kernel/res_counter.c
+++ cgroup-2.6.25-rc2-mm1/kernel/res_counter.c
@@ -92,6 +92,11 @@ ssize_t res_counter_read(struct res_coun
    pos, buf, s - buf);
 }

+u64 res_counter_read_u64(struct res_counter *counter, int member)
+{
+ return *res_counter_member(counter, member);
+}
+
 ssize_t res_counter_write(struct res_counter *counter, int member,
   const char __user *userbuf, size_t nbytes, loff_t *pos,
   int (*write_strategy)(char *st_buf, unsigned long long *val))

--
```

Subject: [PATCH 04/10] CGroup API files: Strip all trailing whitespace in cgroup_write_u64
Posted by Paul Menage on Sat, 23 Feb 2008 22:47:29 GMT
View Forum Message <> Reply to Message

This removes the need for people to remember to pass the -n flag to
echo when writing values to cgroup control files.

Signed-off-by: Paul Menage <menage@google.com>

---
 kernel/cgroup.c | 5 +----
 1 file changed, 1 insertion(+), 4 deletions(-)

Index: cgroup-2.6.25-rc2-mm1/kernel/cgroup.c
===
--- cgroup-2.6.25-rc2-mm1.orig/kernel/cgroup.c
+++ cgroup-2.6.25-rc2-mm1/kernel/cgroup.c
@@ -1321,10 +1321,7 @@ static ssize_t cgroup_write_u64(struct c
   return -EFAULT;

  buffer[nbytes] = 0;   /* nul-terminate */
-
- /* strip newline if necessary */
- if (nbytes && (buffer[nbytes-1] == '\n'))
- buffer[nbytes-1] = 0;
+ strstrip(buffer);
  val = simple_strtoull(buffer, &end, 0);
  if (*end)
   return -EINVAL;

--

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

Subject: [PATCH 05/10] CGroup API files: Update cpusets to use cgroup structured file API
Posted by Paul Menage on Sat, 23 Feb 2008 22:47:30 GMT
View Forum Message <> Reply to Message

Many of the cpusets control files are simple integer values, which
don't require the overhead of memory allocations for reads and writes.

Move the handlers for these control files into cpuset_read_u64() and
cpuset_write_u64().

Signed-off-by: Paul Menage <menage@google.com>

---
 kernel/cpuset.c | 155 ++++++++++++++++++++++++++++++++----------------------
 1 file changed, 81 insertions(+), 74 deletions(-)

Index: cgroup-2.6.25-rc2-mm1/kernel/cpuset.c
===
--- cgroup-2.6.25-rc2-mm1.orig/kernel/cpuset.c
+++ cgroup-2.6.25-rc2-mm1/kernel/cpuset.c
@@ -999,19 +999,6 @@ int current_cpuset_is_being_rebound(void
 }

 /*
- * Call with cgroup_mutex held.
- */
-
-static int update_memory_pressure_enabled(struct cpuset *cs, char *buf)
-{
- if (simple_strtoul(buf, NULL, 10) != 0)
- cpuset_memory_pressure_enabled = 1;
- else
- cpuset_memory_pressure_enabled = 0;
- return 0;
-}
-
-/*
  * update_flag - read a 0 or a 1 in a file and update associated flag
  * bit: the bit to update (CS_CPU_EXCLUSIVE, CS_MEM_EXCLUSIVE,
  *   CS_SCHED_LOAD_BALANCE,
@@ -1023,15 +1010,13 @@ static int update_memory_pressure_enable
  * Call with cgroup_mutex held.
  */

-static int update_flag(cpuset_flagbits_t bit, struct cpuset *cs, char *buf)
+static int update_flag(cpuset_flagbits_t bit, struct cpuset *cs,
+   int turning_on)
 {
- int turning_on;
  struct cpuset trialcs;
  int err;
  int cpus_nonempty, balance_flag_changed;

- turning_on = (simple_strtoul(buf, NULL, 10) != 0);
-
  trialcs = *cs;
  if (turning_on)

```
   set_bit(bit, &trialcs.flags);
@@ -1247,43 +1232,65 @@ static ssize_t cpuset_common_file_write(
  case FILE_MEMLIST:
   retval = update_nodemask(cs, buffer);
   break;
+ default:
+  retval = -EINVAL;
+  goto out2;
+ }
+
+ if (retval == 0)
+  retval = nbytes;
+out2:
+ cgroup_unlock();
+out1:
+ kfree(buffer);
+ return retval;
+}
+
+static int cpuset_write_u64(struct cgroup *cgrp, struct cftype *cft, u64 val)
+{
+ int retval = 0;
+ struct cpuset *cs = cgroup_cs(cgrp);
+ cpuset_filetype_t type = cft->private;
+
+ cgroup_lock();
+
+ if (cgroup_is_removed(cgrp)) {
+  cgroup_unlock();
+  return -ENODEV;
+ }
+
+ switch (type) {
  case FILE_CPU_EXCLUSIVE:
-  retval = update_flag(CS_CPU_EXCLUSIVE, cs, buffer);
+  retval = update_flag(CS_CPU_EXCLUSIVE, cs, val);
   break;
  case FILE_MEM_EXCLUSIVE:
-  retval = update_flag(CS_MEM_EXCLUSIVE, cs, buffer);
+  retval = update_flag(CS_MEM_EXCLUSIVE, cs, val);
   break;
  case FILE_SCHED_LOAD_BALANCE:
-  retval = update_flag(CS_SCHED_LOAD_BALANCE, cs, buffer);
+  retval = update_flag(CS_SCHED_LOAD_BALANCE, cs, val);
   break;
  case FILE_MEMORY_MIGRATE:
-  retval = update_flag(CS_MEMORY_MIGRATE, cs, buffer);
+  retval = update_flag(CS_MEMORY_MIGRATE, cs, val);
```

```
   break;
  case FILE_MEMORY_PRESSURE_ENABLED:
-  retval = update_memory_pressure_enabled(cs, buffer);
+  cpuset_memory_pressure_enabled = !!val;
   break;
  case FILE_MEMORY_PRESSURE:
   retval = -EACCES;
   break;
  case FILE_SPREAD_PAGE:
-  retval = update_flag(CS_SPREAD_PAGE, cs, buffer);
+  retval = update_flag(CS_SPREAD_PAGE, cs, val);
   cs->mems_generation = cpuset_mems_generation++;
   break;
  case FILE_SPREAD_SLAB:
-  retval = update_flag(CS_SPREAD_SLAB, cs, buffer);
+  retval = update_flag(CS_SPREAD_SLAB, cs, val);
   cs->mems_generation = cpuset_mems_generation++;
   break;
  default:
   retval = -EINVAL;
-  goto out2;
+  break;
  }
-
- if (retval == 0)
-  retval = nbytes;
-out2:
  cgroup_unlock();
-out1:
- kfree(buffer);
  return retval;
 }

@@ -1345,30 +1352,6 @@ static ssize_t cpuset_common_file_read(s
  case FILE_MEMLIST:
   s += cpuset_sprintf_memlist(s, cs);
   break;
- case FILE_CPU_EXCLUSIVE:
-  *s++ = is_cpu_exclusive(cs) ? '1' : '0';
-  break;
- case FILE_MEM_EXCLUSIVE:
-  *s++ = is_mem_exclusive(cs) ? '1' : '0';
-  break;
- case FILE_SCHED_LOAD_BALANCE:
-  *s++ = is_sched_load_balance(cs) ? '1' : '0';
-  break;
- case FILE_MEMORY_MIGRATE:
-  *s++ = is_memory_migrate(cs) ? '1' : '0';
```

```
-  break;
- case FILE_MEMORY_PRESSURE_ENABLED:
-  *s++ = cpuset_memory_pressure_enabled ? '1' : '0';
-  break;
- case FILE_MEMORY_PRESSURE:
-  s += sprintf(s, "%d", fmeter_getrate(&cs->fmeter));
-  break;
- case FILE_SPREAD_PAGE:
-  *s++ = is_spread_page(cs) ? '1' : '0';
-  break;
- case FILE_SPREAD_SLAB:
-  *s++ = is_spread_slab(cs) ? '1' : '0';
-  break;
   default:
    retval = -EINVAL;
    goto out;
@@ -1381,8 +1364,32 @@ out:
  return retval;
 }

-
-
+static u64 cpuset_read_u64(struct cgroup *cont, struct cftype *cft)
+{
+ struct cpuset *cs = cgroup_cs(cont);
+ cpuset_filetype_t type = cft->private;
+ switch (type) {
+ case FILE_CPU_EXCLUSIVE:
+  return is_cpu_exclusive(cs);
+ case FILE_MEM_EXCLUSIVE:
+  return is_mem_exclusive(cs);
+ case FILE_SCHED_LOAD_BALANCE:
+  return is_sched_load_balance(cs);
+ case FILE_MEMORY_MIGRATE:
+  return is_memory_migrate(cs);
+ case FILE_MEMORY_PRESSURE_ENABLED:
+  return cpuset_memory_pressure_enabled;
+ case FILE_MEMORY_PRESSURE:
+  return fmeter_getrate(&cs->fmeter);
+  break;
+ case FILE_SPREAD_PAGE:
+  return is_spread_page(cs);
+ case FILE_SPREAD_SLAB:
+  return is_spread_slab(cs);
+ default:
+  BUG();
+ }
+}
```

```
 /*
@@ -1405,57 +1412,57 @@ static struct cftype cft_mems = {

 static struct cftype cft_cpu_exclusive = {
  .name = "cpu_exclusive",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_u64 = cpuset_read_u64,
+ .write_u64 = cpuset_write_u64,
  .private = FILE_CPU_EXCLUSIVE,
 };

 static struct cftype cft_mem_exclusive = {
  .name = "mem_exclusive",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_u64 = cpuset_read_u64,
+ .write_u64 = cpuset_write_u64,
  .private = FILE_MEM_EXCLUSIVE,
 };

 static struct cftype cft_sched_load_balance = {
  .name = "sched_load_balance",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_u64 = cpuset_read_u64,
+ .write_u64 = cpuset_write_u64,
  .private = FILE_SCHED_LOAD_BALANCE,
 };

 static struct cftype cft_memory_migrate = {
  .name = "memory_migrate",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_u64 = cpuset_read_u64,
+ .write_u64 = cpuset_write_u64,
  .private = FILE_MEMORY_MIGRATE,
 };

 static struct cftype cft_memory_pressure_enabled = {
  .name = "memory_pressure_enabled",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_u64 = cpuset_read_u64,
+ .write_u64 = cpuset_write_u64,
  .private = FILE_MEMORY_PRESSURE_ENABLED,
```

```
 };

 static struct cftype cft_memory_pressure = {
  .name = "memory_pressure",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_u64 = cpuset_read_u64,
+ .write_u64 = cpuset_write_u64,
  .private = FILE_MEMORY_PRESSURE,
 };

 static struct cftype cft_spread_page = {
  .name = "memory_spread_page",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_u64 = cpuset_read_u64,
+ .write_u64 = cpuset_write_u64,
  .private = FILE_SPREAD_PAGE,
 };

 static struct cftype cft_spread_slab = {
  .name = "memory_spread_slab",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_u64 = cpuset_read_u64,
+ .write_u64 = cpuset_write_u64,
  .private = FILE_SPREAD_SLAB,
 };

@@ -1584,7 +1591,7 @@ static void cpuset_destroy(struct cgroup
  cpuset_update_task_memory_state();

  if (is_sched_load_balance(cs))
-  update_flag(CS_SCHED_LOAD_BALANCE, cs, "0");
+  update_flag(CS_SCHED_LOAD_BALANCE, cs, 0);

  number_of_cpusets--;
  kfree(cs);

--
```

_____

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

---

Subject: [PATCH 07/10] CGroup API files: Use cgroup map for memcontrol stats file

Remove the seq_file boilerplate used to construct the memcontrol stats
map, and instead use the new map representation for cgroup control
files

Signed-off-by: Paul Menage <menage@google.com>

---
 mm/memcontrol.c | 30 ++++++------------------------
 1 file changed, 6 insertions(+), 24 deletions(-)

Index: cgroup-2.6.25-rc2-mm1/mm/memcontrol.c
===
--- cgroup-2.6.25-rc2-mm1.orig/mm/memcontrol.c
+++ cgroup-2.6.25-rc2-mm1/mm/memcontrol.c
@@ -971,9 +971,9 @@ static const struct mem_cgroup_stat_desc
  [MEM_CGROUP_STAT_RSS] = { "rss", PAGE_SIZE, },
 };

-static int mem_control_stat_show(struct seq_file *m, void *arg)
+static int mem_control_stat_show(struct cgroup *cont, struct cftype *cft,
+   struct cgroup_map_cb *cb)
 {
- struct cgroup *cont = m->private;
  struct mem_cgroup *mem_cont = mem_cgroup_from_cont(cont);
  struct mem_cgroup_stat *stat = &mem_cont->stat;
  int i;
@@ -983,8 +983,7 @@ static int mem_control_stat_show(struct

   val = mem_cgroup_read_stat(stat, i);
   val *= mem_cgroup_stat_desc[i].unit;
- seq_printf(m, "%s %lld\n", mem_cgroup_stat_desc[i].msg,
-    (long long)val);
+ cb->fill(cb, mem_cgroup_stat_desc[i].msg, val);
  }
  /* showing # of active pages */
  {
@@ -994,29 +993,12 @@ static int mem_control_stat_show(struct
       MEM_CGROUP_ZSTAT_INACTIVE);
   active = mem_cgroup_get_all_zonestat(mem_cont,
       MEM_CGROUP_ZSTAT_ACTIVE);
- seq_printf(m, "active %ld\n", (active) * PAGE_SIZE);
- seq_printf(m, "inactive %ld\n", (inactive) * PAGE_SIZE);
+ cb->fill(cb, "active", (active) * PAGE_SIZE);
+ cb->fill(cb, "inactive", (inactive) * PAGE_SIZE);
  }
  return 0;

```
 }

-static const struct file_operations mem_control_stat_file_operations = {
- .read = seq_read,
- .llseek = seq_lseek,
- .release = single_release,
-};
-
-static int mem_control_stat_open(struct inode *unused, struct file *file)
-{
- /* XXX __d_cont */
- struct cgroup *cont = file->f_dentry->d_parent->d_fsdata;
-
- file->f_op = &mem_control_stat_file_operations;
- return single_open(file, mem_control_stat_show, cont);
-}
-
-
-
 static struct cftype mem_cgroup_files[] = {
  {
   .name = "usage_in_bytes",
@@ -1041,7 +1023,7 @@ static struct cftype mem_cgroup_files[]
  },
  {
   .name = "stat",
-  .open = mem_control_stat_open,
+  .read_map = mem_control_stat_show,
  },
 };


--
```

Subject: [PATCH 08/10] CGroup API files: Drop mem_cgroup_force_empty()
Posted by Paul Menage on Sat, 23 Feb 2008 22:47:33 GMT
View Forum Message <> Reply to Message

This function isn't needed - a NULL pointer in the cftype read
function will result in the same EINVAL response to userspace.

Signed-off-by: Paul Menage <menage@google.com>

```
---
 mm/memcontrol.c |   14 --------------
 1 file changed, 14 deletions(-)

Index: cgroup-2.6.25-rc2-mm1/mm/memcontrol.c
===================================================================
--- cgroup-2.6.25-rc2-mm1.orig/mm/memcontrol.c
+++ cgroup-2.6.25-rc2-mm1/mm/memcontrol.c
@@ -950,19 +950,6 @@ static ssize_t mem_force_empty_write(str
  return ret;
 }

-/*
- * Note: This should be removed if cgroup supports write-only file.
- */
-
-static ssize_t mem_force_empty_read(struct cgroup *cont,
-    struct cftype *cft,
-    struct file *file, char __user *userbuf,
-    size_t nbytes, loff_t *ppos)
-{
- return -EINVAL;
-}
-
-
 static const struct mem_cgroup_stat_desc {
  const char *msg;
  u64 unit;
@@ -1019,7 +1006,6 @@ static struct cftype mem_cgroup_files[]
  {
   .name = "force_empty",
   .write = mem_force_empty_write,
-  .read = mem_force_empty_read,
  },
  {
   .name = "stat",

--
```

Subject: [PATCH 09/10] CGroup API files: Move "releasable" to cgroup_debug subsystem

Posted by Paul Menage on Sat, 23 Feb 2008 22:47:34 GMT

The "releasable" control file provided by the cgroup framework exports
the state of a per-cgroup flag that's related to the notify-on-release
feature. This isn't really generally useful, unless you're trying to
debug this particular feature of cgroups.

This patch moves the "releasable" file to the cgroup_debug subsystem.

Signed-off-by: Paul Menage <menage@google.com>

---
 include/linux/cgroup.h | 11 ++++++++++
 kernel/cgroup.c | 23 ----------------------
 kernel/cgroup_debug.c | 12 ++++++++++-
 3 files changed, 22 insertions(+), 24 deletions(-)

Index: cgroup-2.6.25-rc2-mm1/include/linux/cgroup.h
===
--- cgroup-2.6.25-rc2-mm1.orig/include/linux/cgroup.h
+++ cgroup-2.6.25-rc2-mm1/include/linux/cgroup.h
@@ -88,6 +88,17 @@ static inline void css_put(struct cgroup
   __css_put(css);
 }

+/* bits in struct cgroup flags field */
+enum {
+ /* Control Group is dead */
+ CGRP_REMOVED,
+ /* Control Group has previously had a child cgroup or a task,
+ * but no longer (only if CGRP_NOTIFY_ON_RELEASE is set) */
+ CGRP_RELEASABLE,
+ /* Control Group requires release notifications to userspace */
+ CGRP_NOTIFY_ON_RELEASE,
+};
+
 struct cgroup {
   unsigned long flags; /* "unsigned long" so bitops work */

Index: cgroup-2.6.25-rc2-mm1/kernel/cgroup.c
===
--- cgroup-2.6.25-rc2-mm1.orig/kernel/cgroup.c
+++ cgroup-2.6.25-rc2-mm1/kernel/cgroup.c
@@ -119,17 +119,6 @@ static int root_count;
  */
 static int need_forkexit_callback;

-/* bits in struct cgroup flags field */
-enum {

```
- /* Control Group is dead */
- CGRP_REMOVED,
- /* Control Group has previously had a child cgroup or a task,
-  * but no longer (only if CGRP_NOTIFY_ON_RELEASE is set) */
- CGRP_RELEASABLE,
- /* Control Group requires release notifications to userspace */
- CGRP_NOTIFY_ON_RELEASE,
-};
-
 /* convenient tests for these bits */
 inline int cgroup_is_removed(const struct cgroup *cgrp)
 {
@@ -1299,7 +1288,6 @@ enum cgroup_filetype {
  FILE_DIR,
  FILE_TASKLIST,
  FILE_NOTIFY_ON_RELEASE,
- FILE_RELEASABLE,
  FILE_RELEASE_AGENT,
 };

@@ -2169,11 +2157,6 @@ static u64 cgroup_read_notify_on_release
  return notify_on_release(cgrp);
 }

-static u64 cgroup_read_releasable(struct cgroup *cgrp, struct cftype *cft)
-{
- return test_bit(CGRP_RELEASABLE, &cgrp->flags);
-}
-
 /*
  * for the common functions, 'private' gives the type of file
  */
@@ -2193,12 +2176,6 @@ static struct cftype files[] = {
   .write = cgroup_common_file_write,
   .private = FILE_NOTIFY_ON_RELEASE,
  },
-
- {
-  .name = "releasable",
-  .read_u64 = cgroup_read_releasable,
-  .private = FILE_RELEASABLE,
- }
 };

 static struct cftype cft_release_agent = {
Index: cgroup-2.6.25-rc2-mm1/kernel/cgroup_debug.c
===================================================================
--- cgroup-2.6.25-rc2-mm1.orig/kernel/cgroup_debug.c
```

```
+++ cgroup-2.6.25-rc2-mm1/kernel/cgroup_debug.c
@@ -1,5 +1,5 @@
 /*
- * kernel/ccontainer_debug.c - Example cgroup subsystem that
+ * kernel/cgroup_debug.c - Example cgroup subsystem that
  * exposes debug info
  *
  * Copyright (C) Google Inc, 2007
@@ -62,6 +62,11 @@ static u64 current_css_set_refcount_read
  return count;
 }

+static u64 releasable_read(struct cgroup *cgrp, struct cftype *cft)
+{
+ return test_bit(CGRP_RELEASABLE, &cgrp->flags);
+}
+
 static struct cftype files[] =  {
  {
   .name = "cgroup_refcount",
@@ -81,6 +86,11 @@ static struct cftype files[] =  {
   .name = "current_css_set_refcount",
   .read_u64 = current_css_set_refcount_read,
  },
+
+ {
+  .name = "releasable",
+  .read_u64 = releasable_read,
+ }
 };

 static int debug_populate(struct cgroup_subsys *ss, struct cgroup *cont)

--
```

Subject: Re: [PATCH 00/10] CGroup API files: Various cleanup to CGroup control files
Posted by Li Zefan on Tue, 26 Feb 2008 03:23:25 GMT
View Forum Message <> Reply to Message

menage@google.com wrote:
> This patchset is a roll-up of the non-contraversial items of the
> various patches that I've sent out recently, fixed according to the

> feedback received.
>
> In summary they are:
>
> - general rename of read_uint/write_uint to read_u64/write_u64
>
> - use these methods for cpusets and memory controller files
>
> - add a read_map cgroup file method, and use it in the memory
> controller
>
> - move the "releasable" control file to the debug subsystem
>
> - make the debug subsystem config option default to "n"
>
> The only user-visible changes are the movement of the "releasable"
> file and the fact that some write_u64()-based control files are now
> more forgiving of additional whitespace at the end of their input.
>
> Signed-off-by: Paul Menage <menage@google.com>
>
> --
> --

Should those pathces be rebased againt 2.6.25-rc3 ?

Subject: Re: [PATCH 00/10] CGroup API files: Various cleanup to CGroup control files

Posted by Paul Menage on Tue, 26 Feb 2008 06:19:28 GMT

View Forum Message <> Reply to Message

On Mon, Feb 25, 2008 at 7:23 PM, Li Zefan <lizf@cn.fujitsu.com> wrote:
>
> Should those pathces be rebased againt 2.6.25-rc3 ?
>

No, because they're against 2.6.25-rc2-mm1, which is already has (I
think) any of the new bits in 2.6.25-rc3 that would be affected by
these patches.

Paul

Subject: Re: [PATCH 00/10] CGroup API files: Various cleanup to CGroup control files
Posted by Li Zefan on Tue, 26 Feb 2008 07:48:50 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On Mon, Feb 25, 2008 at 7:23 PM, Li Zefan <lizf@cn.fujitsu.com> wrote:
>> Should those pathces be rebased againt 2.6.25-rc3 ?
>>
>
> No, because they're against 2.6.25-rc2-mm1, which is already has (I
> think) any of the new bits in 2.6.25-rc3 that would be affected by
> these patches.
>
> Paul

-rc2-mm1 came out on 2008-02-16, but the patches I posted several days ago
has been merged into -rc3, so your patches don't apply now. :(

Think about ./MAINTAINERS update and set up a git tree for development of
cgroup and cgroup subsystems as Andrew suggested. :)

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

Subject: Re: [PATCH 05/10] CGroup API files: Update cpusets to use cgroup structured file API
Posted by Li Zefan on Wed, 27 Feb 2008 00:54:08 GMT
View Forum Message <> Reply to Message

menage@google.com wrote:
> + case FILE_MEMORY_MIGRATE:
> + return is_memory_migrate(cs);
> + case FILE_MEMORY_PRESSURE_ENABLED:
> + return cpuset_memory_pressure_enabled;
> + case FILE_MEMORY_PRESSURE:
> + return fmeter_getrate(&cs->fmeter);
> + break;

redundant 'break'

Subject: Re: [PATCH 00/10] CGroup API files: Various cleanup to CGroup control files
Posted by Paul Menage on Wed, 27 Feb 2008 01:09:17 GMT
View Forum Message <> Reply to Message

On Mon, Feb 25, 2008 at 11:48 PM, Li Zefan <lizf@cn.fujitsu.com> wrote:
>
> -rc2-mm1 came out on 2008-02-16, but the patches I posted several days ago
> has been merged into -rc3, so your patches don't apply now. :(

OK, good point.

>
> Think about ./MAINTAINERS update and set up a git tree for development of
> cgroup and cgroup subsystems as Andrew suggested. :)
>

Sounds like a good plan - I'll get on to it ...

Paul

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

Subject: Re: [PATCH 05/10] CGroup API files: Update cpusets to use cgroup structured file API
Posted by Paul Menage on Wed, 27 Feb 2008 01:09:48 GMT
View Forum Message <> Reply to Message

On Tue, Feb 26, 2008 at 4:54 PM, Li Zefan <lizf@cn.fujitsu.com> wrote:
> menage@google.com wrote:
> > > + case FILE_MEMORY_MIGRATE:
> > > + return is_memory_migrate(cs);
> > > + case FILE_MEMORY_PRESSURE_ENABLED:
> > > + return cpuset_memory_pressure_enabled;
> > > + case FILE_MEMORY_PRESSURE:
> > > + return fmeter_getrate(&cs->fmeter);
> > > + break;
>
> redundant 'break'

>

Thanks, I'll zap that.

Paul

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