Subject: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Paul Menage on Fri, 15 Feb 2008 20:44:25 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_uint() and
cpuset_write_uint(). This also has the advantage that the control
files show up as "u64" rather than "string" in the cgroup.api file.

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

---
 kernel/cpuset.c | 158 ++++++++++++++++++++++++++++++++----------------------
 1 file changed, 83 insertions(+), 75 deletions(-)

Index: cgroupmap-2.6.24-mm1/kernel/cpuset.c
===
--- cgroupmap-2.6.24-mm1.orig/kernel/cpuset.c
+++ cgroupmap-2.6.24-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);
@@ -1247,44 +1232,66 @@ 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_uint(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;
+ return -EINVAL;
 }

 /*
@@ -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_uint(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 +1412,58 @@ 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_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
  .private = FILE_CPU_EXCLUSIVE,
 };

 static struct cftype cft_mem_exclusive = {
  .name = "mem_exclusive",
- .read = cpuset_common_file_read,
- .write = cpuset_common_file_write,
+ .read_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
  .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_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
  .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_uint = cpuset_read_uint,
+ .read_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
  .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_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
  .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_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
  .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_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
  .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_uint = cpuset_read_uint,
+ .write_uint = cpuset_write_uint,
  .private = FILE_SPREAD_SLAB,
 };

@@ -1584,7 +1592,7 @@ 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: Re: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Paul Jackson on Sun, 17 Feb 2008 03:29:01 GMT
View Forum Message <> Reply to Message

Ok ... this would (I suspect, just from code reading, no bytes were harmed in actual testing of this) have a minor change to how white space is handled writing integer flags to cpuset files, and a minor inconstency.

1) Existing cpuset code lets you set a flag (e.g. cpu_exclusive) by doing:
echo '1 rumplestiltskin' > cpu_exclusive # same as: echo 1 > cpu_exclusive
   With this patch, that probably fails, EINVAL.

2) With this patch, one can write "1" or "1\n" to cpuset integer files, but one
   cannot successfully write "1\r\n" or "1 " or "1 \n". However, for the cpuset
   control files that take strings, not single integers, one -can- have any mix
   of trailing white space.

So far as I know, I have no requirement to write rumplestiltskin to cpuset files ;).
So I'm content to let the minor change in (1) pass without further comment.

I'd like to recommend consideration of the following patch, to address the minor inconsistency of (2), and to save a few bytes of kernel text space.

=======

From: Paul Jackson <pj@sgi.com>

Strip all trailing whitespace (such as carriage returns)
when parsing integer writes to cgroup files, not just
one trailing newline if present.

Signed-off-by: Paul Jackson <pj@sgi.com>
Cc: Paul Menage <menage@google.com>

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

--- 2.6.24-mm1.orig/kernel/cgroup.c 2008-02-16 04:20:33.000000000 -0800
+++ 2.6.24-mm1/kernel/cgroup.c 2008-02-16 19:00:41.207478218 -0800
@@ -1321,10 +1321,7 @@ static ssize_t cgroup_write_uint(struct
   return -EFAULT;

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

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.940.382.4214

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

---

Subject: Re: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Paul Menage on Sun, 17 Feb 2008 17:18:44 GMT
View Forum Message <> Reply to Message

On Feb 16, 2008 7:29 PM, Paul Jackson <pj@sgi.com> wrote:
>
> From: Paul Jackson <pj@sgi.com>
>
> Strip all trailing whitespace (such as carriage returns)
> when parsing integer writes to cgroup files, not just
> one trailing newline if present.

Sounds like a good idea to me. Thanks for this.

>
> Signed-off-by: Paul Jackson <pj@sgi.com>
> Cc: Paul Menage <menage@google.com>

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

>
> ---
> kernel/cgroup.c | 5 +----
> 1 file changed, 1 insertion(+), 4 deletions(-)
>
> --- 2.6.24-mm1.orig/kernel/cgroup.c   2008-02-16 04:20:33.000000000 -0800
> +++ 2.6.24-mm1/kernel/cgroup.c 2008-02-16 19:00:41.207478218 -0800
> @@ -1321,10 +1321,7 @@ static ssize_t cgroup_write_uint(struct
>   return -EFAULT;
>
>   buffer[nbytes] = 0;   /* nul-terminate */
> -
> -   /* strip newline if necessary */
> -   if (nbytes && (buffer[nbytes-1] == '\n'))
> -   buffer[nbytes-1] = 0;
> +   strstrip(buffer);   /* strip -just- trailing whitespace */
>   val = simple_strtoull(buffer, &end, 0);
>   if (*end)
>   return -EINVAL;
>
>
> --
>   I won't rest till it's the best ...
>   Programmer, Linux Scalability
>   Paul Jackson <pj@sgi.com> 1.940.382.4214
>

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

---

Subject: Re: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Paul Jackson on Sun, 17 Feb 2008 17:28:43 GMT
View Forum Message <> Reply to Message

> > Strip all trailing whitespace (such as carriage returns)
> > when parsing integer writes to cgroup files, not just
> > one trailing newline if present.
>
> Sounds like a good idea to me. Thanks for this.

I'm figuring it would be easiest if you just threw this
little change into your hopper for the bigger changes

you're making ... no need to preserve my name as author
of this.

But if you'd rather you or I send this separately to
Andrew Morton, that works too.

--

I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.940.382.4214

_____

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

---

Subject: Re: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Paul Menage on Sun, 17 Feb 2008 17:48:56 GMT
View Forum Message <> Reply to Message

On Feb 17, 2008 9:28 AM, Paul Jackson <pj@sgi.com> wrote:
>
> I'm figuring it would be easiest if you just threw this
> little change into your hopper for the bigger changes
> you're making

OK, will do.

Paul

_____

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

---

Subject: Re: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Li Zefan on Mon, 18 Feb 2008 09:55:27 GMT
View Forum Message <> Reply to Message

Paul Jackson wrote:
> Ok ... this would (I suspect, just from code reading, no bytes were
> harmed in actual testing of this) have a minor change to how white
> space is handled writing integer flags to cpuset files, and a minor
> inconstency.
>

> 1) Existing cpuset code lets you set a flag (e.g. cpu_exclusive) by doing:
> echo '1 rumplestiltskin' > cpu_exclusive   # same as: echo 1 > cpu_exclusive
>   With this patch, that probably fails, EINVAL.
>
> 2) With this patch, one can write "1" or "1\n" to cpuset integer files, but one
>   cannot successfully write "1\r\n" or "1 " or "1 \n". However, for the cpuset
>   control files that take strings, not single integers, one -can- have any mix
>   of trailing white space.
>
> So far as I know, I have no requirement to write rumplestiltskin to cpuset files ;).
> So I'm content to let the minor change in (1) pass without further comment.
>
> I'd like to recommend consideration of the following patch, to address the
> minor inconsistency of (2), and to save a few bytes of kernel text space.
>

For memory controller, we have to do this:
 echo -n 4m > memory.limit_in_bytes
'-n' is necessary. This is another inconsistency..

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

---

Subject: Re: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Balbir Singh on Mon, 18 Feb 2008 11:13:33 GMT
View Forum Message <> Reply to Message

* Balbir Singh <balbir@linux.vnet.ibm.com> [2008-02-18 16:42:05]:

> Li Zefan wrote:
> > Paul Jackson wrote:
> >> Ok ... this would (I suspect, just from code reading, no bytes were
> >> harmed in actual testing of this) have a minor change to how white
> >> space is handled writing integer flags to cpuset files, and a minor
> >> inconstency.
> >>
> >> 1) Existing cpuset code lets you set a flag (e.g. cpu_exclusive) by doing:
> >> echo '1 rumplestiltskin' > cpu_exclusive   # same as: echo 1 > cpu_exclusive
> >>   With this patch, that probably fails, EINVAL.
> >>
> >> 2) With this patch, one can write "1" or "1\n" to cpuset integer files, but one
> >>   cannot successfully write "1\r\n" or "1 " or "1 \n". However, for the cpuset
> >>   control files that take strings, not single integers, one -can- have any mix
> >>   of trailing white space.
> >>

> >> So far as I know, I have no requirement to write rumplestiltskin to cpuset files ;).
> >> So I'm content to let the minor change in (1) pass without further comment.
> >>
> >> I'd like to recommend consideration of the following patch, to address the
> >> minor inconsistency of (2), and to save a few bytes of kernel text space.
> >>
> >
> > For memory controller, we have to do this:
> > echo -n 4m > memory.limit_in_bytes
> > '-n' is necessary. This is another inconsistency..
>
Hi. Li,

I have a similar patch that fixes the inconsistency.
It's attached below. Andrew, could we please consider this patch for -mm

The memory controller has a requirement that while writing values, we need
to use echo -n. This patch fixes the problem and makes the UI more consistent.

Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>

 Documentation/controllers/memory.txt | 8 ++++----
 kernel/res_counter.c | 1 +
 2 files changed, 5 insertions(+), 4 deletions(-)

diff -puN mm/memcontrol.c~memory-controller-fix-crlf-echo-issue mm/memcontrol.c
diff -puN kernel/res_counter.c~memory-controller-fix-crlf-echo-issue kernel/res_counter.c
--- linux-2.6.25-rc2/kernel/res_counter.c~memory-controller-fix-crlf-echo-issue 2008-02-18
16:15:02.000000000 +0530
+++ linux-2.6.25-rc2-balbir/kernel/res_counter.c 2008-02-18 16:16:16.000000000 +0530
@@ -113,6 +113,7 @@ ssize_t res_counter_write(struct res_cou

  ret = -EINVAL;

+ strstrip(buf);
  if (write_strategy) {
   if (write_strategy(buf, &tmp)) {
    goto out_free;
diff -puN Documentation/controllers/memory.txt~memory-controller-fix-crlf-echo-issue
Documentation/controllers/memory.txt
---
linux-2.6.25-rc2/Documentation/controllers/memory.txt~memory-controller-fix-crlf-echo-issue 2008
-02-18 16:18:26.000000000 +0530
+++ linux-2.6.25-rc2-balbir/Documentation/controllers/memory.txt 2008-02-18
16:18:44.000000000 +0530
@@ -164,7 +164,7 @@ c. Enable CONFIG_CGROUP_MEM_CONT

Since now we're in the 0 cgroup,
We can alter the memory limit:
-# echo -n 4M > /cgroups/0/memory.limit_in_bytes
+# echo 4M > /cgroups/0/memory.limit_in_bytes

NOTE: We can use a suffix (k, K, m, M, g or G) to indicate values in kilo,
mega or gigabytes.
@@ -185,7 +185,7 @@ number of factors, such as rounding up t
availability of memory on the system. The user is required to re-read
this file after a write to guarantee the value committed by the kernel.

-# echo -n 1 > memory.limit_in_bytes
+# echo 1 > memory.limit_in_bytes
cat memory.limit_in_bytes
4096 Bytes

@@ -197,7 +197,7 @@ caches, RSS and Active pages/Inactive pa

The memory.force_empty gives an interface to drop *all* charges by force.

-# echo -n 1 > memory.force_empty
+# echo 1 > memory.force_empty

will drop all charges in cgroup. Currently, this is maintained for test.

@@ -238,7 +238,7 @@ rmdir() if there are no tasks.
The type of memory accounted by the cgroup can be limited to just
mapped pages by writing "1" to memory.control_type field

-echo -n 1 > memory.control_type
+echo > memory.control_type

5. TODO

_

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

Subject: Re: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Andreas Schwab on Mon, 18 Feb 2008 11:52:12 GMT
View Forum Message <> Reply to Message

Balbir Singh <balbir@linux.vnet.ibm.com> writes:

> @@ -238,7 +238,7 @@ rmdir() if there are no tasks.
> The type of memory accounted by the cgroup can be limited to just
> mapped pages by writing "1" to memory.control_type field
>
> -echo -n 1 > memory.control_type
> +echo > memory.control_type

Looks like you stripped too much here.

Andreas.

--
Andreas Schwab, SuSE Labs, schwab@suse.de

PGP key fingerprint = 58CA 54C7 6D53 942B 1756 01D3 44D5 214B 8276 4ED5
"And now for something completely different."

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

Subject: Re: [RFC][PATCH 7/7] CGroup API: Update cpusets to use cgroup structured file API
Posted by Balbir Singh on Mon, 18 Feb 2008 11:56:25 GMT
View Forum Message <> Reply to Message

* Balbir Singh <balbir@linux.vnet.ibm.com> [2008-02-18 17:23:42]:

> Andreas Schwab wrote:
> > Balbir Singh <balbir@linux.vnet.ibm.com> writes:
> >
> >> @@ -238,7 +238,7 @@ rmdir() if there are no tasks.
> >> The type of memory accounted by the cgroup can be limited to just
> >> mapped pages by writing "1" to memory.control_type field
> >>
> >> -echo -n 1 > memory.control_type
> >> +echo > memory.control_type
> >
> > Looks like you stripped too much here.
> >

> > Andreas.
> >
>
Yikes,

The control type feature documentation needs to go away and Li has a patch for
it, so I'll not touch that part. Here's the updated patch. Thanks for catching
this Andreas.

The memory controller has a requirement that while writing values, we need
to use echo -n. This patch fixes the problem and makes the UI more consistent.

Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>

 Documentation/controllers/memory.txt | 6 +++---
 kernel/res_counter.c | 1 +
 2 files changed, 4 insertions(+), 3 deletions(-)

diff -puN mm/memcontrol.c~memory-controller-fix-crlf-echo-issue mm/memcontrol.c
diff -puN kernel/res_counter.c~memory-controller-fix-crlf-echo-issue kernel/res_counter.c
--- linux-2.6.25-rc2/kernel/res_counter.c~memory-controller-fix-crlf-echo-issue 2008-02-18
16:15:02.000000000 +0530
+++ linux-2.6.25-rc2-balbir/kernel/res_counter.c 2008-02-18 16:16:16.000000000 +0530
@@ -113,6 +113,7 @@ ssize_t res_counter_write(struct res_cou

  ret = -EINVAL;

+ strstrip(buf);
  if (write_strategy) {
   if (write_strategy(buf, &tmp)) {
    goto out_free;
diff -puN Documentation/controllers/memory.txt~memory-controller-fix-crlf-echo-issue
Documentation/controllers/memory.txt
---
linux-2.6.25-rc2/Documentation/controllers/memory.txt~memory-controller-fix-crlf-echo-issue 2008
-02-18 16:18:26.000000000 +0530
+++ linux-2.6.25-rc2-balbir/Documentation/controllers/memory.txt 2008-02-18
17:24:48.000000000 +0530
@@ -164,7 +164,7 @@ c. Enable CONFIG_CGROUP_MEM_CONT

 Since now we're in the 0 cgroup,
 We can alter the memory limit:
-# echo -n 4M > /cgroups/0/memory.limit_in_bytes
+# echo 4M > /cgroups/0/memory.limit_in_bytes

 NOTE: We can use a suffix (k, K, m, M, g or G) to indicate values in kilo,
 mega or gigabytes.

@@ -185,7 +185,7 @@ number of factors, such as rounding up t
 availability of memory on the system. The user is required to re-read
 this file after a write to guarantee the value committed by the kernel.

-# echo -n 1 > memory.limit_in_bytes
+# echo 1 > memory.limit_in_bytes
 # cat memory.limit_in_bytes
 4096 Bytes

@@ -197,7 +197,7 @@ caches, RSS and Active pages/Inactive pa

 The memory.force_empty gives an interface to drop *all* charges by force.

-# echo -n 1 > memory.force_empty
+# echo 1 > memory.force_empty

 will drop all charges in cgroup. Currently, this is maintained for test.

_

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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