

[View Forum Message](#) <> [Reply to Message](#)

```

bool "Include legacy /proc/<pid>/cpuset file"
depends on CPUSETS
diff -puN mm/Makefile~mem-control-setup mm/Makefile
--- linux-2.6.22-rc6/mm/Makefile~mem-control-setup 2007-07-04 15:05:22.000000000 -0700
+++ linux-2.6.22-rc6-balbir/mm/Makefile 2007-07-04 15:05:22.000000000 -0700
@@ -30,4 +30,5 @@ obj-$(CONFIG_FS_XIP) += filemap_xip.o
obj-$(CONFIG_MIGRATION) += migrate.o
obj-$(CONFIG_SMP) += allocpercpu.o
obj-$(CONFIG_QUICKLIST) += quicklist.o
+obj-$(CONFIG_CONTAINER_MEM_CONT) += memcontrol.o

diff -puN /dev/null mm/memcontrol.c
--- /dev/null 2007-06-01 08:12:04.000000000 -0700
+++ linux-2.6.22-rc6-balbir/mm/memcontrol.c 2007-07-04 15:05:22.000000000 -0700
@@ -0,0 +1,141 @@
+/* memcontrol.c - Memory Controller
+ *
+ * Copyright IBM Corporation, 2007
+ * Author Balbir Singh <balbir@linux.vnet.ibm.com>
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of version 2.1 of the GNU Lesser General Public License
+ * as published by the Free Software Foundation.
+ *
+ * This program is distributed in the hope that it would be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
+ */
+
+#include <linux/res_counter.h>
+#include <linux/memcontrol.h>
+#include <linux/container.h>
+
+struct container_subsys mem_container_subsys;
+
+/*
+ * The memory controller data structure. The memory controller controls both
+ * page cache and RSS per container. We would eventually like to provide
+ * statistics based on the statistics developed by Rik Van Riel for clock-pro,
+ * to help the administrator determine what knobs to tune.
+ *
+ * TODO: Add a water mark for the memory controller. Reclaim will begin when
+ * we hit the water mark.
+ */
+struct mem_container {
+ struct container_subsys_state css;
+ /*
+  * the counter to account for memory usage

```

```

+ */
+ struct res_counter res;
+};
+
+/*
+ * A meta page is associated with every page descriptor. The meta page
+ * helps us identify information about the container
+ */
+struct meta_page {
+ struct list_head list; /* per container LRU list */
+ struct page *page;
+ struct mem_container *mem_container;
+};
+
+
+static inline struct mem_container *mem_container_from_cont(struct container
+    *cnt)
+{
+ return container_of(container_subsys_state(cnt,
+ mem_container_subsys_id), struct mem_container,
+ css);
+}
+
+static ssize_t mem_container_read(struct container *cont, struct cftype *cft,
+ struct file *file, char __user *userbuf, size_t nbytes,
+ loff_t *ppos)
+{
+ return res_counter_read(&mem_container_from_cont(cont)->res,
+ cft->private, userbuf, nbytes, ppos);
+}
+
+static ssize_t mem_container_write(struct container *cont, struct cftype *cft,
+ struct file *file, const char __user *userbuf,
+ size_t nbytes, loff_t *ppos)
+{
+ return res_counter_write(&mem_container_from_cont(cont)->res,
+ cft->private, userbuf, nbytes, ppos);
+}
+
+static struct cftype mem_container_usage = {
+ .name = "mem_usage",
+ .private = RES_USAGE,
+ .read = mem_container_read,
+};
+
+static struct cftype mem_container_limit = {
+ .name = "mem_limit",
+ .private = RES_LIMIT,

```

```

+ .write = mem_container_write,
+ .read = mem_container_read,
+};
+
+static struct cftype mem_container_failcnt = {
+ .name = "mem_failcnt",
+ .private = RES_FAILCNT,
+ .read = mem_container_read,
+};
+
+static int mem_container_create(struct container_subsys *ss,
+ struct container *cont)
+{
+ struct mem_container *mem;
+
+ mem = kzalloc(sizeof(struct mem_container), GFP_KERNEL);
+ if (!mem)
+ return -ENOMEM;
+
+ res_counter_init(&mem->res);
+ cont->subsys[mem_container_subsys_id] = &mem->css;
+ mem->css.container = cont;
+ return 0;
+}
+
+static void mem_container_destroy(struct container_subsys *ss,
+ struct container *cont)
+{
+ kfree(mem_container_from_cont(cont));
+}
+
+static int mem_container_populate(struct container_subsys *ss,
+ struct container *cont)
+{
+ int rc = 0;
+
+ rc = container_add_file(cont, &mem_container_usage);
+ if (rc < 0)
+ goto err;
+
+ rc = container_add_file(cont, &mem_container_limit);
+ if (rc < 0)
+ goto err;
+
+ rc = container_add_file(cont, &mem_container_failcnt);
+ if (rc < 0)
+ goto err;
+
+

```

