Subject: [PATCH 1/4] - v2 - Provide a new procfs interface to set next id
Posted by Nadia Derbey on Fri, 18 Apr 2008 05:45:00 GMT
View Forum Message <> Reply to Message

[PATCH 01/04]

This patch proposes the procfs facilities needed to feed the id for the
next object to be allocated.

if an
echo "LONG XX" > /proc/self/task/<tid>/next_id
is issued, next object to be created will have XX as its id.

This applies to objects that need a single id, such as ipc objects.

Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

---
 fs/exec.c | 3 +
 fs/proc/base.c | 76 ++
 include/linux/sched.h | 3 +
 include/linux/sysids.h | 24 ++++++++++++++++
 kernel/Makefile | 2 -
 kernel/exit.c | 4 ++
 kernel/fork.c | 2 +
 kernel/nextid.c | 86 ++
 8 files changed, 199 insertions(+), 1 deletion(-)

Index: linux-2.6.25-rc8-mm2/include/linux/sysids.h
===
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.25-rc8-mm2/include/linux/sysids.h 2008-04-17 13:35:29.000000000 +0200
@@ -0,0 +1,24 @@
+/*
+ * include/linux/sysids.h
+ *
+ * Definitions to support object creation with predefined id.
+ *
+ */
+
+#ifndef _LINUX_SYSIDS_H
+#define _LINUX_SYSIDS_H
+
+struct sys_id {
+ long id;
+};
+
+extern ssize_t get_nextid(struct task_struct *, char *, size_t);

```
+extern int set_nextid(struct task_struct *, char *);
+extern int reset_nextid(struct task_struct *);
+
+static inline void exit_nextid(struct task_struct *tsk)
+{
+ reset_nextid(tsk);
+}
+
+#endif /* _LINUX_SYSIDS_H */
Index: linux-2.6.25-rc8-mm2/include/linux/sched.h
===================================================================
--- linux-2.6.25-rc8-mm2.orig/include/linux/sched.h 2008-04-17 12:50:22.000000000 +0200
+++ linux-2.6.25-rc8-mm2/include/linux/sched.h 2008-04-17 13:38:04.000000000 +0200
@@ -88,6 +88,7 @@ struct sched_param {
 #include <linux/task_io_accounting.h>
 #include <linux/kobject.h>
 #include <linux/latencytop.h>
+#include <linux/sysids.h>

 #include <asm/processor.h>

@@ -1280,6 +1281,8 @@ struct task_struct {
  int latency_record_count;
  struct latency_record latency_record[LT_SAVECOUNT];
 #endif
+ /* Id to assign to the next resource to be created */
+ struct sys_id *next_id;
 };

 /*
Index: linux-2.6.25-rc8-mm2/fs/proc/base.c
===================================================================
--- linux-2.6.25-rc8-mm2.orig/fs/proc/base.c 2008-04-17 12:50:40.000000000 +0200
+++ linux-2.6.25-rc8-mm2/fs/proc/base.c 2008-04-17 15:19:25.000000000 +0200
@@ -1138,6 +1138,77 @@ static const struct file_operations proc
 #endif


+static ssize_t next_id_read(struct file *file, char __user *buf,
+    size_t count, loff_t *ppos)
+{
+ struct task_struct *task;
+ char *page;
+ ssize_t length;
+
+ task = get_proc_task(file->f_path.dentry->d_inode);
+ if (!task)
+  return -ESRCH;
```

```
+
+ if (count >= PAGE_SIZE)
+  count = PAGE_SIZE - 1;
+
+ length = -ENOMEM;
+ page = (char *) __get_free_page(GFP_TEMPORARY);
+ if (!page)
+  goto out;
+
+ length = get_nextid(task, (char *) page, count);
+ if (length >= 0)
+  length = simple_read_from_buffer(buf, count, ppos,
+      (char *)page, length);
+ free_page((unsigned long) page);
+
+out:
+ put_task_struct(task);
+ return length;
+}
+
+static ssize_t next_id_write(struct file *file, const char __user *buf,
+    size_t count, loff_t *ppos)
+{
+ struct inode *inode = file->f_path.dentry->d_inode;
+ char *page;
+ ssize_t length;
+
+ if (pid_task(proc_pid(inode), PIDTYPE_PID) != current)
+  return -EPERM;
+
+ if (count >= PAGE_SIZE)
+  count = PAGE_SIZE - 1;
+
+ if (*ppos != 0) {
+  /* No partial writes. */
+  return -EINVAL;
+ }
+ page = (char *)__get_free_page(GFP_TEMPORARY);
+ if (!page)
+  return -ENOMEM;
+ length = -EFAULT;
+ if (copy_from_user(page, buf, count))
+  goto out_free_page;
+
+ page[count] = '\0';
+
+ length = set_nextid(current, page);
+ if (!length)
```

```
+  length = count;
+
+out_free_page:
+ free_page((unsigned long) page);
+ return length;
+}
+
+static const struct file_operations proc_next_id_operations = {
+ .read  = next_id_read,
+ .write  = next_id_write,
+};
+
+
 #ifdef CONFIG_SCHED_DEBUG
 /*
  * Print out various scheduling related per-task fields:
@@ -2779,6 +2850,11 @@ static const struct pid_entry tid_base_s
 #ifdef CONFIG_FAULT_INJECTION
  REG("make-it-fail", S_IRUGO|S_IWUSR, fault_inject),
 #endif
+ /*
+  * NOTE that next_id is not added into tgid_base_stuff[] since it has
+  * to be specified on a per-thread basis.
+  */
+ REG("next_id", S_IRUGO|S_IWUSR, next_id),
 };

 static int proc_tid_base_readdir(struct file * filp,
Index: linux-2.6.25-rc8-mm2/kernel/Makefile
===================================================================
--- linux-2.6.25-rc8-mm2.orig/kernel/Makefile 2008-04-17 12:50:25.000000000 +0200
+++ linux-2.6.25-rc8-mm2/kernel/Makefile 2008-04-17 13:47:45.000000000 +0200
@@ -9,7 +9,7 @@ obj-y     = sched.o fork.o exec_domain.o
      rcupdate.o extable.o params.o posix-timers.o \
      kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
      hrtimer.o rwsem.o nsproxy.o srcu.o semaphore.o \
-     notifier.o ksysfs.o pm_qos_params.o
+     notifier.o ksysfs.o pm_qos_params.o nextid.o

 ifdef CONFIG_FTRACE
 # Do not profile debug utilities
Index: linux-2.6.25-rc8-mm2/kernel/nextid.c
===================================================================
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.25-rc8-mm2/kernel/nextid.c 2008-04-17 15:16:07.000000000 +0200
@@ -0,0 +1,86 @@
+/*
+ * linux/kernel/nextid.c
```

```
+ *
+ *
+ * Provide the get_nextid() / set_nextid() routines
+ * (called from fs/proc/base.c).
+ * They allow to specify the id for the next resource to be allocated,
+ * instead of letting the allocator set it for us.
+ */
+
+#include <linux/sched.h>
+#include <linux/ctype.h>
+
+
+
+ssize_t get_nextid(struct task_struct *task, char *buffer, size_t size)
+{
+ struct sys_id *sid;
+
+ sid = task->next_id;
+ if (!sid)
+  return snprintf(buffer, size, "UNSET\n");
+
+ return snprintf(buffer, size, "LONG %ld\n", sid->id);
+}
+
+static int set_single_id(struct task_struct *task, char *buffer)
+{
+ struct sys_id *sid;
+ long next_id;
+ char *end;
+
+ next_id = simple_strtol(buffer, &end, 0);
+ if (end == buffer || (end && *end && !isspace(*end)))
+  return -EINVAL;
+
+ sid = task->next_id;
+ if (!sid) {
+  sid = kzalloc(sizeof(*sid), GFP_KERNEL);
+  if (!sid)
+   return -ENOMEM;
+  task->next_id = sid;
+ }
+
+ sid->id = next_id;
+
+ return 0;
+}
+
+int reset_nextid(struct task_struct *task)
```

```
+{
+ struct sys_id *sid;
+
+ sid = task->next_id;
+ if (!sid)
+  return 0;
+
+ task->next_id = NULL;
+ kfree(sid);
+ return 0;
+}
+
+#define LONG_STR    "LONG"
+#define RESET_STR   "RESET"
+
+/*
+ * Parses a line written to /proc/self/task/<my_tid>/next_id.
+ * this line has the following format:
+ * LONG id              --> a single id is specified
+ */
+int set_nextid(struct task_struct *task, char *buffer)
+{
+ char *token, *out = buffer;
+
+ if (!out)
+  return -EINVAL;
+
+ token = strsep(&out, " ");
+
+ if (!strcmp(token, LONG_STR))
+  return set_single_id(task, out);
+ else if (!strncmp(token, RESET_STR, strlen(RESET_STR)))
+  return reset_nextid(task);
+ else
+  return -EINVAL;
+}
Index: linux-2.6.25-rc8-mm2/kernel/fork.c
===================================================================
--- linux-2.6.25-rc8-mm2.orig/kernel/fork.c 2008-04-17 12:50:25.000000000 +0200
+++ linux-2.6.25-rc8-mm2/kernel/fork.c 2008-04-17 13:49:09.000000000 +0200
@@ -1167,6 +1167,8 @@ static struct task_struct *copy_process(
  p->blocked_on = NULL; /* not blocked yet */
 #endif

+ p->next_id = NULL;
+
  /* Perform scheduler related setup. Assign this task to a CPU. */
  sched_fork(p, clone_flags);
```

```
Index: linux-2.6.25-rc8-mm2/kernel/exit.c
===================================================================
--- linux-2.6.25-rc8-mm2.orig/kernel/exit.c 2008-04-17 12:50:25.000000000 +0200
+++ linux-2.6.25-rc8-mm2/kernel/exit.c 2008-04-17 13:50:42.000000000 +0200
@@ -987,6 +987,10 @@ NORET_TYPE void do_exit(long code)

  proc_exit_connector(tsk);
  exit_notify(tsk, group_dead);
+
+ if (unlikely(tsk->next_id))
+  exit_nextid(tsk);
+
 #ifdef CONFIG_NUMA
  mpol_put(tsk->mempolicy);
  tsk->mempolicy = NULL;
Index: linux-2.6.25-rc8-mm2/fs/exec.c
===================================================================
--- linux-2.6.25-rc8-mm2.orig/fs/exec.c 2008-04-17 12:50:41.000000000 +0200
+++ linux-2.6.25-rc8-mm2/fs/exec.c 2008-04-17 13:51:47.000000000 +0200
@@ -1024,6 +1024,9 @@ int flush_old_exec(struct linux_binprm *
  flush_signal_handlers(current, 0);
  flush_old_files(current->files);

+ if (unlikely(current->next_id))
+  reset_nextid(current);
+
  return 0;

 mmap_failed:

--
```