
Subject: [RFC][PATCH 1/4] Provide a new procfs interface to set next ipc id
Posted by [Nadia Derby](#) on Mon, 10 Mar 2008 13:50:55 GMT
[View Forum Message](#) <> [Reply to Message](#)

[PATCH 01/04]

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

if an
echo XX > /proc/self/next_ipcid
is issued, next ipc object to be created will have XX as its id.

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

fs/proc/base.c | 66 +++
include/linux/sched.h | 3 ++
include/linux/sysids.h | 20 +++++
kernel/Makefile | 2 -
kernel/set_nextid.c | 31 +++++
5 files changed, 121 insertions(+), 1 deletion(-)

Index: linux-2.6.25-rc3-mm1/include/linux/sysids.h

=====

--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.25-rc3-mm1/include/linux/sysids.h 2008-03-10 11:39:10.000000000 +0100
@@ -0,0 +1,20 @@
+/*
+ * include/linux/sysids.h
+ *
+ * Definitions to support object creation with predefined id.
+ *
+ */
+
+#ifndef _LINUX_SYSIDS_H
+#define _LINUX_SYSIDS_H
+
+#define SYS_ID_IPC 1
+
+struct sys_id {
+ int flag; /* which id should be set */
+ int ipc;
+};
+
+extern int ipc_set_nextid(struct task_struct *, int id);
+
+#endif /* _LINUX_SYSIDS_H */

Index: linux-2.6.25-rc3-mm1/include/linux/sched.h

```
=====
--- linux-2.6.25-rc3-mm1.orig/include/linux/sched.h 2008-03-10 09:18:46.000000000 +0100
+++ linux-2.6.25-rc3-mm1/include/linux/sched.h 2008-03-10 09:28:30.000000000 +0100
@@ -87,6 +87,7 @@ 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>
```

```
@@ -1261,6 +1262,8 @@ 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-rc3-mm1/fs/proc/base.c

```
=====
--- linux-2.6.25-rc3-mm1.orig/fs/proc/base.c 2008-03-10 09:19:39.000000000 +0100
+++ linux-2.6.25-rc3-mm1/fs/proc/base.c 2008-03-10 11:22:20.000000000 +0100
@@ -76,6 +76,7 @@
#include <linux/oom.h>
#include <linux/elf.h>
#include <linux/pid_namespace.h>
+#include <linux/ctype.h>
#include "internal.h"
```

```
/* NOTE:
@@ -1080,6 +1081,69 @@ static const struct file_operations proc
#endif
```

```
+static ssize_t next_ipcid_read(struct file *file, char __user *buf,
+    size_t count, loff_t *ppos)
+{
+ struct task_struct *task;
+ char buffer[PROC_NUMBUF];
+ size_t len;
+ struct sys_id *sid;
+ int next_ipcid;
+
+ task = get_proc_task(file->f_path.dentry->d_inode);
+ if (!task)
```

```

+ return -ESRCH;
+
+ sid = task->next_id;
+ next_ipcid = (sid) ? ((sid->flag & SYS_ID_IPC) ? sid->ipc : -1)
+ : -1;
+
+ put_task_struct(task);
+
+ len = snprintf(buffer, sizeof(buffer), "%i\n", next_ipcid);
+
+ return simple_read_from_buffer(buf, count, ppos, buffer, len);
+}
+
+static ssize_t next_ipcid_write(struct file *file, const char __user *buf,
+ size_t count, loff_t *ppos)
+{
+ struct inode *inode = file->f_path.dentry->d_inode;
+ char buffer[PROC_NUMBUF], *end;
+ int next_ipcid;
+ int rc;
+
+ if (pid_task(proc_pid(inode), PIDTYPE_PID) != current)
+ return -EPERM;
+
+ memset(buffer, 0, sizeof(buffer));
+ if (count > sizeof(buffer) - 1)
+ count = sizeof(buffer) - 1;
+ if (copy_from_user(buffer, buf, count))
+ return -EFAULT;
+
+ next_ipcid = simple_strtol(buffer, &end, 0);
+ if (next_ipcid < 0 || end == buffer)
+ return -EINVAL;
+
+ while (isspace(*end))
+ end++;
+
+ rc = ipc_set_nextid(current, next_ipcid);
+ if (rc)
+ return rc;
+
+ if (end - buffer == 0)
+ return -EIO;
+ return end - buffer;
+}
+
+static const struct file_operations proc_next_ipcid_operations = {
+ .read = next_ipcid_read,

```

```

+ .write = next_ipcid_write,
+};
+
+
#ifdef CONFIG_SCHED_DEBUG
/*
 * Print out various scheduling related per-task fields:
@@ -2391,6 +2455,7 @@ static const struct pid_entry tgid_base_
#ifdef CONFIG_TASK_IO_ACCOUNTING
    INF("io", S_IRUGO, pid_io_accounting),
#endif
+ REG("next_ipcid", S_IRUGO|S_IWUSR, next_ipcid),
};

```

```

static int proc_tgid_base_readdir(struct file * filp,
@@ -2716,6 +2781,7 @@ static const struct pid_entry tid_base_s
#ifdef CONFIG_FAULT_INJECTION
    REG("make-it-fail", S_IRUGO|S_IWUSR, fault_inject),
#endif
+ REG("next_ipcid", S_IRUGO|S_IWUSR, next_ipcid),
};

```

```

static int proc_tid_base_readdir(struct file * filp,

```

Index: linux-2.6.25-rc3-mm1/kernel/Makefile

```

=====
--- linux-2.6.25-rc3-mm1.orig/kernel/Makefile 2008-03-10 09:19:01.000000000 +0100
+++ linux-2.6.25-rc3-mm1/kernel/Makefile 2008-03-10 09:41:27.000000000 +0100
@@ -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 set_nexttid.o

```

```

obj-$(CONFIG_SYSCTL_SYSCALL_CHECK) += sysctl_check.o

```

```

obj-$(CONFIG_STACKTRACE) += stacktrace.o

```

Index: linux-2.6.25-rc3-mm1/kernel/set_nexttid.c

```

=====
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.25-rc3-mm1/kernel/set_nexttid.c 2008-03-10 10:09:47.000000000 +0100
@@ -0,0 +1,31 @@
+/*
+ * linux/kernel/set_nexttid.c
+ *
+ * Provide the XXX_set_nexttid() 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>  
+  
+  
+  
+int ipc_set_nextid(struct task_struct *task, int id)  
+{  
+ struct sys_id *sid;  
+  
+ sid = task->next_id;  
+ if (!sid) {  
+ sid = kzalloc(sizeof(*sid), GFP_KERNEL);  
+ if (!sid)  
+ return -ENOMEM;  
+ task->next_id = sid;  
+ }  
+  
+ sid->ipc = id;  
+ sid->flag |= SYS_ID_IPC;  
+  
+ return 0;  
+}  
+
```