
Subject: [RFC PATCH 2/6] IPC/sem: per <pid> semundo file in procfs
Posted by [Nadia Derby](#) on Wed, 25 Jun 2008 13:49:12 GMT
[View Forum Message](#) <> [Reply to Message](#)

PATCH [02/06]

This patch adds a new procfs interface to display the per-process semundo data.

A new per-PID file is added, called "semundo".

It contains one line per semaphore IPC where there is something to undo for this process.

Then, each line contains the semid followed by each undo value corresponding to each semaphores of the semaphores array.

This interface will be particularly useful to allow a user access these data, for example for checkpointing a process

With this patch, the semundo file can only be accessed in read mode.

When this file is opened, if an undo_list exists for the target process, it is accessed in an rcu read section, and its refcount is incremented, avoiding that it be freed.

The reverse is done during the release operation, and the undo_list is freed if the process reading the file was the last process accessing that list.

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>

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

```
fs/proc/base.c    | 3 +
fs/proc/internal.h | 1
ipc/sem.c         | 127 +++++
3 files changed, 128 insertions(+), 3 deletions(-)
```

Index: linux-2.6.26-rc5-mm3/fs/proc/base.c

```
=====
--- linux-2.6.26-rc5-mm3.orig/fs/proc/base.c 2008-06-24 09:05:09.000000000 +0200
+++ linux-2.6.26-rc5-mm3/fs/proc/base.c 2008-06-24 10:03:33.000000000 +0200
@@ -2525,6 +2525,9 @@ static const struct pid_entry tgid_base_
#ifdef CONFIG_TASK_IO_ACCOUNTING
    INF("io", S_IRUGO, tgid_io_accounting),
#endif
+#ifdef CONFIG_SYSVIPC
+    REG("semundo", S_IRUGO, semundo),
+#endif
};
```

```
static int proc_tgid_base_readdir(struct file * filp,
```

```
Index: linux-2.6.26-rc5-mm3/fs/proc/internal.h
```

```
--- linux-2.6.26-rc5-mm3.orig/fs/proc/internal.h 2008-06-24 09:05:09.000000000 +0200
```

```
+++ linux-2.6.26-rc5-mm3/fs/proc/internal.h 2008-06-24 10:04:19.000000000 +0200
```

```
@@ -65,6 +65,7 @@ extern const struct file_operations proc
extern const struct file_operations proc_net_operations;
extern const struct file_operations proc_kmsg_operations;
extern const struct inode_operations proc_net_inode_operations;
+extern const struct file_operations proc_semundo_operations;
```

```
void free_proc_entry(struct proc_dir_entry *de);
```

```
Index: linux-2.6.26-rc5-mm3/ipc/sem.c
```

```
--- linux-2.6.26-rc5-mm3.orig/ipc/sem.c 2008-06-24 09:37:33.000000000 +0200
```

```
+++ linux-2.6.26-rc5-mm3/ipc/sem.c 2008-06-24 10:59:46.000000000 +0200
```

```
@@ -97,6 +97,7 @@ static void freeary(struct ipc_namespace
#ifdef CONFIG_PROC_FS
static int sysvipc_sem_proc_show(struct seq_file *s, void *it);
#endif
```

```
+static void free_semundo_list(struct sem_undo_list *, struct ipc_namespace *);
```

```
#define SEMMSL_FAST 256 /* 512 bytes on stack */
```

```
#define SEMOPM_FAST 64 /* ~ 372 bytes on stack */
```

```
@@ -1282,8 +1283,14 @@ void exit_sem(struct task_struct *tsk)
    rcu_assign_pointer(tsk->sysvsem.undo_list, NULL);
    synchronize_rcu();
```

```
- if (!atomic_dec_and_test(&ulp->refcnt))
```

```
- return;
```

```
+ if (atomic_dec_and_test(&ulp->refcnt))
```

```
+ free_semundo_list(ulp, tsk->nsproxy->ipc_ns);
```

```
+}
```

```
+
```

```
+static void free_semundo_list(struct sem_undo_list *ulp,
```

```
+ struct ipc_namespace *ipc_ns)
```

```
+{
```

```
+ BUG_ON(atomic_read(&ulp->refcnt));
```

```
for (;;) {
```

```
    struct sem_array *sma;
```

```
@@ -1303,7 +1310,7 @@ void exit_sem(struct task_struct *tsk)
```

```
    if (semid == -1)
```

```
        break;
```

```
- sma = sem_lock_check(tsk->nsproxy->ipc_ns, un->semid);
```

```
+ sma = sem_lock_check(ipc_ns, un->semid);
```

```

/* exit_sem raced with IPC_RMID, nothing to do */
if (IS_ERR(sma))
@@ -1384,4 +1391,118 @@ static int sysvipc_sem_proc_show(struct
    sma->sem_otime,
    sma->sem_ctime);
}
+
+struct undo_list_data {
+ struct sem_undo_list *undo_list;
+ struct ipc_namespace *ipc_ns;
+};
+
+/* iterator */
+static void *semundo_start(struct seq_file *m, loff_t *ppos)
+{
+ return NULL;
+}
+
+static void *semundo_next(struct seq_file *m, void *v, loff_t *ppos)
+{
+ return NULL;
+}
+
+static void semundo_stop(struct seq_file *m, void *v)
+{
+ return;
+}
+
+static int semundo_show(struct seq_file *m, void *v)
+{
+ return 0;
+}
+
+static struct seq_operations semundo_op = {
+ .start = semundo_start,
+ .next = semundo_next,
+ .stop = semundo_stop,
+ .show = semundo_show
+};
+
+static struct sem_undo_list *get_proc_ulp(struct task_struct *tsk)
+{
+ struct sem_undo_list *ulp;
+
+ rcu_read_lock();
+ ulp = rcu_dereference(tsk->sysvsem.undo_list);
+ if (ulp)

```

```

+ if (!atomic_inc_not_zero(&ulp->refcnt))
+   ulp = NULL;
+ rcu_read_unlock();
+ return ulp;
+}
+
+static void put_proc_ulp(struct sem_undo_list *ulp,
+   struct ipc_namespace *ns)
+{
+ if (ulp && atomic_dec_and_test(&ulp->refcnt))
+   free_semundo_list(ulp, ns);
+}
+
+/*
+ * semundo_open: open operation for /proc/<PID>/semundo file
+ */
+static int semundo_open(struct inode *inode, struct file *file)
+{
+ struct task_struct *task;
+ struct sem_undo_list *ulp;
+ struct undo_list_data *data;
+ struct ipc_namespace *ns;
+ int ret = 0;
+
+ data = kzalloc(sizeof(*data), GFP_KERNEL);
+ if (!data)
+   return -ENOMEM;
+
+ task = get_pid_task(PROC_I(inode)->pid, PIDTYPE_PID);
+ if (!task) {
+   ret = -EINVAL;
+   goto out_err;
+ }
+
+ ulp = get_proc_ulp(task);
+ ns = get_ipc_ns(task->nsproxy->ipc_ns);
+ put_task_struct(task);
+
+ ret = seq_open(file, &semundo_op);
+ if (!ret) {
+   struct seq_file *m = file->private_data;
+   data->undo_list = ulp;
+   data->ipc_ns = ns;
+   m->private = data;
+   return 0;
+ }
+
+ put_proc_ulp(ulp, ns);

```

```

+ put_ipc_ns(ns);
+out_err:
+ kfree(data);
+ return ret;
+}
+
+static int semundo_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *m = file->private_data;
+ struct undo_list_data *data = m->private;
+ struct sem_undo_list *ulp = data->undo_list;
+ struct ipc_namespace *ns = data->ipc_ns;
+
+ put_proc_ulp(ulp, ns);
+ put_ipc_ns(ns);
+ kfree(data);
+ return seq_release(inode, file);
+}
+
+const struct file_operations proc_semundo_operations = {
+ .open = semundo_open,
+ .read = seq_read,
+ .llseek = seq_lseek,
+ .release = semundo_release,
+};
#endif

--

```

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