Subject: [RFC PATCH 0/6] SYSVIPC/semaphores - allow saving/restoring a process' semundo_list
Posted by Nadia Derbey on Wed, 25 Jun 2008 13:49:10 GMT
View Forum Message <> Reply to Message

This patchset is a part of an effort to make sysv ipc objects
read/writable from userspace for checkpoint / restart.

System V ipc's are objects that are global to a system and can thus be
checkpointed and restarted on a container basis. But some parts of the ipc
structures are process related and, as such should be checkpointed / restarted
on a process basis.

Message queues needn't and thus cannot be reached from a task structure
(only the other direction is possible: a task can be reached by a msg receiver
or sender structure if it is sleeping).

Shared memories are accessible from a task structure through that task's
memory mapping (/proc/<pid>/maps shows a process' memory maps).

Semaphores are kind of accessible from a task structure too: the task
structure's sysvsem field makes it possible to walk through all the semaphores
operations to undo when a process is exiting.
This list, that need to be saved and restored during a process' c/r,
cannot yet be accessed from user space.

This is a feature that will be needed if ever we take the direction of driving
checkpoint / restart from user space, though the read part of it could be used
even from now on.

Since this undo_list is, again, on a thread basis, we propose to externalize
it via a new proc file: /proc/<pid>/semundo.

Actually, Pierre Pieffer has already done the proposal in threads
https://lists.linux-foundation.org/pipermail/containers/2008-January/thread.html#9756
up to #9759

I've ported them to 2.6.26-rc5-mm3, and I'm now coming back with a
simpler implementation: the write operation is now only allowed into
/proc/self/semundo, which simplifies the locking strategy.

These patches should be applied to 2.6.26-rc5-mm3, in the following order:
[PATCH 1/6] : ipc_rcu_protect_access_to_undo_list.patch
Makes the process' undo_list rcu protected in order to enable
safely reading it.
[PATCH 2/6] : ipc_procfs_semundo_file.patch
Introduces the semundo proc file (the seq operations are still

empty).
[PATCH 3/6] : ipc_procfs_semundo_start_stop_seqops.patch
Introduces the .start and .stop seq operations.
[PATCH 4/6] : ipc_procfs_semundo_next_seqop.patch
Introduces the .next seq operation.
[PATCH 5/6] : ipc_procfs_semundo_show_seqop.patch
Introduces the .show seq operation.
[PATCH 6/6] : ipc_procfs_semundo_write.patch
The semundo proc file becomes writable.

Comments are welcome!

Regards,
Nadia

--

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

---

Subject: [RFC PATCH 2/6] IPC/sem: per <pid> semundo file in procfs
Posted by Nadia Derbey 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 Derbey <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

---

Subject: [RFC PATCH 3/6] IPC/sem: start/stop operations for /proc/pid/semundo
Posted by Nadia Derbey on Wed, 25 Jun 2008 13:49:13 GMT
View Forum Message <> Reply to Message

PATCH [03/06]

This patch introduces the .start and .stop seq operations for
/proc/pid/semundo.

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

---
 ipc/sem.c | 43 ++---
 1 file changed, 41 insertions(+), 2 deletions(-)

Index: linux-2.6.26-rc5-mm3/ipc/sem.c
===
--- linux-2.6.26-rc5-mm3.orig/ipc/sem.c 2008-06-24 10:59:46.000000000 +0200
+++ linux-2.6.26-rc5-mm3/ipc/sem.c 2008-06-24 12:32:36.000000000 +0200
@@ -1400,7 +1400,42 @@ struct undo_list_data {
 /* iterator */
 static void *semundo_start(struct seq_file *m, loff_t *ppos)
 {
- return NULL;
+ struct undo_list_data *data = m->private;
+ struct sem_undo_list *ulp = data->undo_list;
+ struct sem_undo *undo;
+ loff_t pos = *ppos;
+
+ if (!ulp)
+ return NULL;
+
+ if (pos < 0)
+ return NULL;
+
+ /* If ulp is not NULL, it means that we've successfully grabbed
+ * a refcnt in semundo_open. That prevents the undo_list from being
+ * freed.
+ *

```
+  * Note:
+  * 1) the sem_undo structure is RCU-protected. So take the rcu read
+  *    lock and only release it during the .stop operation.
+  * 2) we accept to release the undo_list lock (i.e. we allow the list
+  *    to change) between each iteration, instead of taking that lock
+  *    during the .start and releasing it during the .stop operation.
+  *    This is to reduce the performance impact on the access to the
+  *    undo_list.
+  */
+ rcu_read_lock();
+ spin_lock(&ulp->lock);
+ list_for_each_entry_rcu(undo, &ulp->list_proc, list_proc) {
+  if ((undo->semid != -1) && !(pos--))
+   break;
+ }
+ spin_unlock(&ulp->lock);
+
+ if (&undo->list_proc == &ulp->list_proc)
+  return NULL;
+
+ return undo;
 }

 static void *semundo_next(struct seq_file *m, void *v, loff_t *ppos)
@@ -1410,7 +1445,11 @@ static void *semundo_next(struct seq_fil

 static void semundo_stop(struct seq_file *m, void *v)
 {
- return;
+ struct undo_list_data *data = m->private;
+ struct sem_undo_list *ulp = data->undo_list;
+
+ if (ulp)
+  rcu_read_unlock();
 }

 static int semundo_show(struct seq_file *m, void *v)

--
```

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

Subject: Re: [RFC PATCH 2/6] IPC/sem: per <pid> semundo file in procfs
Posted by serue on Wed, 25 Jun 2008 20:33:37 GMT

Quoting Nadia.Derbey@bull.net (Nadia.Derbey@bull.net):
> 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 Derbey <Nadia.Derbey@bull.net>

Acked-by: Serge Hallyn <serue@us.ibm.com>

-serge

>
> ---
> 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

---

Subject: Re: [RFC PATCH 3/6] IPC/sem: start/stop operations for /proc/pid/semundo
Posted by serue on Wed, 25 Jun 2008 20:39:50 GMT
View Forum Message <> Reply to Message

Quoting Nadia.Derbey@bull.net (Nadia.Derbey@bull.net):
> PATCH [03/06]
>
> This patch introduces the .start and .stop seq operations for
> /proc/pid/semundo.
>

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

Acked-by: Serge Hallyn <serue@us.ibm.com>

> ---
> ipc/sem.c | 43 ++---
> 1 file changed, 41 insertions(+), 2 deletions(-)
>
> Index: linux-2.6.26-rc5-mm3/ipc/sem.c
> ===
> --- linux-2.6.26-rc5-mm3.orig/ipc/sem.c 2008-06-24 10:59:46.000000000 +0200
> +++ linux-2.6.26-rc5-mm3/ipc/sem.c 2008-06-24 12:32:36.000000000 +0200
> @@ -1400,7 +1400,42 @@ struct undo_list_data {
> /* iterator */
> static void *semundo_start(struct seq_file *m, loff_t *ppos)
> {
> - return NULL;
> + struct undo_list_data *data = m->private;
> + struct sem_undo_list *ulp = data->undo_list;
> + struct sem_undo *undo;
> + loff_t pos = *ppos;
> +
> + if (!ulp)
> + return NULL;
> +
> + if (pos < 0)
> + return NULL;
> +
> + /* If ulp is not NULL, it means that we've successfully grabbed
> + * a refcnt in semundo_open. That prevents the undo_list from being
> + * freed.
> + *
> + * Note:
> + * 1) the sem_undo structure is RCU-protected. So take the rcu read
> + *   lock and only release it during the .stop operation.
> + * 2) we accept to release the undo_list lock (i.e. we allow the list
> + *   to change) between each iteration, instead of taking that lock
> + *   during the .start and releasing it during the .stop operation.
> + *   This is to reduce the performance impact on the access to the
> + *   undo_list.
> + */
> + rcu_read_lock();
> + spin_lock(&ulp->lock);
> + list_for_each_entry_rcu(undo, &ulp->list_proc, list_proc) {
> + if ((undo->semid != -1) && !(pos--))
> +   break;
> + }

> + spin_unlock(&ulp->lock);
> +
> + if (&undo->list_proc == &ulp->list_proc)
> + return NULL;
> +
> + return undo;
> }
>
> static void *semundo_next(struct seq_file *m, void *v, loff_t *ppos)
> @@ -1410,7 +1445,11 @@ static void *semundo_next(struct seq_fil
>
> static void semundo_stop(struct seq_file *m, void *v)
> {
> - return;
> + struct undo_list_data *data = m->private;
> + struct sem_undo_list *ulp = data->undo_list;
> +
> + if (ulp)
> + rcu_read_unlock();
> }
>
> static int semundo_show(struct seq_file *m, void *v)
>
> --

_____

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

---

Subject: Re: [RFC PATCH 2/6] IPC/sem: per <pid> semundo file in procfs
Posted by Michael Kerrisk on Thu, 26 Jun 2008 05:08:14 GMT
View Forum Message <> Reply to Message

On 6/25/08, Nadia.Derbey@bull.net <Nadia.Derbey@bull.net> wrote:
> 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 Derbey <Nadia.Derbey@bull.net>

Nadia,

This is a kernel-userland interface change, please do CC me, so that I
can change the man pages if needed.

Cheers,

Michael

> ---
> 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
>
```

--
Michael Kerrisk
Linux man-pages maintainer; http://www.kernel.org/doc/man-pages/
Found a bug? http://www.kernel.org/doc/man-pages/reporting_bugs.html

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

---

Subject: Re: [RFC PATCH 2/6] IPC/sem: per <pid> semundo file in procfs
Posted by Nadia Derbey on Thu, 26 Jun 2008 06:07:45 GMT
View Forum Message <> Reply to Message

Michael Kerrisk wrote:
> On 6/25/08, Nadia.Derbey@bull.net <Nadia.Derbey@bull.net> wrote:
>
>>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 Derbey <Nadia.Derbey@bull.net>
>
>
> Nadia,
>
> This is a kernel-userland interface change, please do CC me, so that I
> can change the man pages if needed.

So sorry Michael, but looks like I forgot the complete documentation
part: I also have an issue from Serge about a missing documentation in
the Documentation directory.
Will fix that and take that opportunity to put you in the loop.

Regards,
Nadia

>
> Cheers,
>
> Michael
>
>
>
>
>> ---
>> 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
>>
>
>
>

--
==
Name..... Nadia DERBEY
Organization.. BULL/DT/OSwR&D/Linux

Email..... mailto:Nadia.Derbey@bull.net
Address..... BULL, B.P. 208, 38432 Echirolles Cedex, France
Tel..... (33) 76 29 77 62 [Internal Bull: (229) 77 62]
Telex,Fax..... 980648 F - (33) 76 29 76 00
Internal Bull. Mail: FREC-B1208
==

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