

Subject: Re: [PATCH] Rework /proc/locks via seq_files and seq_list helpers
Posted by [Pavel Emelianov](#) on Thu, 20 Sep 2007 08:27:32 GMT

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

J. Bruce Fields wrote:

> On Wed, Sep 19, 2007 at 03:35:27PM +0400, Pavel Emelyanov wrote:

```
>> Currently /proc/locks is shown with a proc_read function, but
>> its behavior is rather complex as it has to manually handle
>> current offset and buffer length. On the other hand, files
>> that show objects from lists can be easily reimplemented using
>> the sequential files and the seq_list XXX() helpers.
```

```
>>
>> This saves (as usually) 16 lines of code and more than 200 from
>> the .text section.
```

```
>>
>> This patch looks rather ugly, as diff often uses curly braces
>> as not-changed lines, but I haven't managed to organize the
>> code to make diff look better. Except for move the whole proc
>> related stuff upper/lower in the locks.c file...
```

```
>
> Fine by me. I've lost track--are you assuming some earlier patches are
> applied? It doesn't seem to apply to any copy of locks.c I have.
```

Yes, this applies the the set of patches that cleanup the MANDATORY_LOCK macro (5 patches just accepted by Andrew).

```
> --b.  
>  
>> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>  
>>  
>> ---  
>>  
>> fs/locks.c      | 124 ++++++-----  
>> fs/proc/proc_misc.c | 20 ++++----  
>> 2 files changed, 64 insertions(+), 80 deletions(-)  
>>  
>> diff --git a/fs/locks.c b/fs/locks.c  
>> index 8e849ed..746dc70 100644  
>> --- a/fs/locks.c  
>> +++ b/fs/locks.c  
>> @@ -2032,134 +2032,116 @@ int vfs_cancel_lock(struct file *filp, s  
>>  
>> EXPORT_SYMBOL_GPL(vfs_cancel_lock);  
>>  
>> -static void lock_get_status(char* out, struct file_lock *fl, int id, char *pfx)  
>> +#ifdef CONFIG_PROC_FS  
>> +#include <linux/seq_file.h>  
>> +
```

```

>> +static void lock_get_status(struct seq_file *f, struct file_lock *fl,
>> +     int id, char *pfx)
>> {
>>     struct inode *inode = NULL;
>>
>>     if (fl->fl_file != NULL)
>>         inode = fl->fl_file->f_path.dentry->d_inode;
>>
>> - out += sprintf(out, "%d:%s ", id, pfx);
>> + seq_printf(f, "%d:%s ", id, pfx);
>>     if (IS_POSIX(fl)) {
>> - out += sprintf(out, "%6s %s ",
>> + seq_printf(f, "%6s %s ",
>>         (fl->fl_flags & FL_ACCESS) ? "ACCESS" : "POSIX ",
>>         (inode == NULL) ? "*NOINODE*" :
>>         mandatory_lock(inode) ? "MANDATORY" : "ADVISORY ");
>>     } else if (IS_FLOCK(fl)) {
>>         if (fl->fl_type & LOCK_MAND) {
>> - out += sprintf(out, "FLOCK MSNFS  ");
>> + seq_printf(f, "FLOCK MSNFS  ");
>>         } else {
>> - out += sprintf(out, "FLOCK ADVISORY ");
>> + seq_printf(f, "FLOCK ADVISORY ");
>>         }
>>     } else if (IS_LEASE(fl)) {
>> - out += sprintf(out, "LEASE ");
>> + seq_printf(f, "LEASE ");
>>         if (fl->fl_type & F_INPROGRESS)
>> - out += sprintf(out, "BREAKING ");
>> + seq_printf(f, "BREAKING ");
>>         else if (fl->fl_file)
>> - out += sprintf(out, "ACTIVE  ");
>> + seq_printf(f, "ACTIVE  ");
>>         else
>> - out += sprintf(out, "BREAKER ");
>> + seq_printf(f, "BREAKER ");
>>     } else {
>> - out += sprintf(out, "UNKNOWN UNKNOWN ");
>> + seq_printf(f, "UNKNOWN UNKNOWN ");
>>     }
>>     if (fl->fl_type & LOCK_MAND) {
>> - out += sprintf(out, "%s ",
>> + seq_printf(f, "%s ",
>>         (fl->fl_type & LOCK_READ)
>>         ? (fl->fl_type & LOCK_WRITE) ? "RW  " : "READ "
>>         : (fl->fl_type & LOCK_WRITE) ? "WRITE" : "NONE ");
>>     } else {
>> - out += sprintf(out, "%s ",

```

```

>> + seq_printf(f, "%s ",
>>         (fl->fl_type & F_INPROGRESS)
>>         ? (fl->fl_type & F_UNLCK) ? "UNLCK" : "READ "
>>         : (fl->fl_type & F_WRLCK) ? "WRITE" : "READ ");
>> }
>> if (inode) {
>> #ifdef WE_CAN_BREAK_LSLK_NOW
>> - out += sprintf(out, "%d %s:%ld ", fl->fl_pid,
>> + seq_printf(f, "%d %s:%ld ", fl->fl_pid,
>>     inode->i_sb->s_id, inode->i_ino);
>> #else
>> /* userspace relies on this representation of dev_t ;-( */
>> - out += sprintf(out, "%d %02x:%02x:%ld ", fl->fl_pid,
>> + seq_printf(f, "%d %02x:%02x:%ld ", fl->fl_pid,
>>     MAJOR(inode->i_sb->s_dev),
>>     MINOR(inode->i_sb->s_dev), inode->i_ino);
>> #endif
>> } else {
>> - out += sprintf(out, "%d <none>:0 ", fl->fl_pid);
>> + seq_printf(f, "%d <none>:0 ", fl->fl_pid);
>> }
>> if (IS_POSIX(fl)) {
>>     if (fl->fl_end == OFFSET_MAX)
>> - out += sprintf(out, "%Ld EOF\n", fl->fl_start);
>> + seq_printf(f, "%Ld EOF\n", fl->fl_start);
>>     else
>> - out += sprintf(out, "%Ld %Ld\n", fl->fl_start,
>> -     fl->fl_end);
>> + seq_printf(f, "%Ld %Ld\n", fl->fl_start, fl->fl_end);
>> } else {
>> - out += sprintf(out, "0 EOF\n");
>> + seq_printf(f, "0 EOF\n");
>> }
>> }
>>
>> -static void move_lock_status(char **p, off_t* pos, off_t offset)
>> +static int locks_show(struct seq_file *f, void *v)
>> {
>> - int len;
>> - len = strlen(*p);
>> - if(*pos >= offset) {
>> - /* the complete line is valid */
>> - *p += len;
>> - *pos += len;
>> - return;
>> - }
>> - if(*pos+len > offset) {
>> - /* use the second part of the line */

```

```

>> - int i = offset-*pos;
>> - memmove(*p,*p+i,len-i);
>> - *p += len-i;
>> - *pos += len;
>> - return;
>> - }
>> - /* discard the complete line */
>> - *pos += len;
>> - }
>> + int idx;
>> + struct file_lock *fl, *bfl;
>>
>> -/**
>> - * get_locks_status - reports lock usage in /proc/locks
>> - * @buffer: address in userspace to write into
>> - * @start: ?
>> - * @offset: how far we are through the buffer
>> - * @length: how much to read
>> - */
>> + fl = list_entry(v, struct file_lock, fl_link);
>> + idx = (int)f->private;
>>
>> -int get_locks_status(char *buffer, char **start, off_t offset, int length)
>> -{
>> - struct file_lock *fl;
>> - char *q = buffer;
>> - off_t pos = 0;
>> - int i = 0;
>> + lock_get_status(f, fl, idx, "");
>>
>> - lock_kernel();
>> - list_for_each_entry(fl, &file_lock_list, fl_link) {
>> - struct file_lock *bfl;
>> + list_for_each_entry(bfl, &fl->fl_block, fl_block)
>> + lock_get_status(f, bfl, idx, " ->");
>>
>> - lock_get_status(q, fl, ++i, "");
>> - move_lock_status(&q, &pos, offset);
>> + f->private = (void *) (idx + 1);
>> + return 0;
>> + }
>>
>> - if(pos >= offset+length)
>> - goto done;
>> +static void *locks_start(struct seq_file *f, loff_t *pos)
>> +{
>> + lock_kernel();
>> + f->private = (void *)1;

```

```

>> + return seq_list_start(&file_lock_list, *pos);
>> +}
>>
>> - list_for_each_entry(bfl, &fl->fl_block, fl_block) {
>> - lock_get_status(q, bfl, i, "->");
>> - move_lock_status(&q, &pos, offset);
>> +static void *locks_next(struct seq_file *f, void *v, loff_t *pos)
>> +{
>> + return seq_list_next(v, &file_lock_list, pos);
>> +}
>>
>> - if(pos >= offset+length)
>> - goto done;
>> - }
>> - }
>> -done:
>> +static void locks_stop(struct seq_file *f, void *v)
>> +{
>> unlock_kernel();
>> - *start = buffer;
>> - if(q-buffer < length)
>> - return (q-buffer);
>> - return length;
>> }
>>
>> +struct seq_operations locks_seq_operations = {
>> + .start = locks_start,
>> + .next = locks_next,
>> + .stop = locks_stop,
>> + .show = locks_show,
>> +};
>> +#endif
>> +
>> /**
>> * lock_may_read - checks that the region is free of locks
>> * @inode: the inode that is being read
>> diff --git a/fs/proc/proc_misc.c b/fs/proc/proc_misc.c
>> index 166a6db..043621c 100644
>> --- a/fs/proc/proc_misc.c
>> +++ b/fs/proc/proc_misc.c
>> @@ -68,7 +68,6 @@ extern int get_stam_list(char *);
>> extern int get_filesystem_list(char *);
>> extern int get_exec_domain_list(char *);
>> extern int get_dma_list(char *);
>> -extern int get_locks_status (char *, char **, off_t, int);
>>
>> static int proc_calc_metrics(char *page, char **start, off_t off,
>> int count, int *eof, int len)

```

```

>> @@ -630,16 +629,19 @@ static int cmdline_read_proc(char *page,
>> return proc_calc_metrics(page, start, off, count, eof, len);
>> }
>>
>> -static int locks_read_proc(char *page, char **start, off_t off,
>> -    int count, int *eof, void *data)
>> +extern struct seq_operations locks_seq_operations;
>> +static int locks_open(struct inode *inode, struct file *filp)
>> {
>> - int len = get_locks_status(page, start, off, count);
>> -
>> - if (len < count)
>> - *eof = 1;
>> - return len;
>> + return seq_open(filp, &locks_seq_operations);
>> }
>>
>> +static const struct file_operations proc_locks_operations = {
>> + .open = locks_open,
>> + .read = seq_read,
>> + .llseek = seq_lseek,
>> + .release = seq_release,
>> +};
>> +
>> static int execdomains_read_proc(char *page, char **start, off_t off,
>>     int count, int *eof, void *data)
>> {
>> @@ -916,7 +918,6 @@ void __init proc_misc_init(void)
>> #endif
>> {"filesystems", filesystems_read_proc},
>> {"cmdline", cmdline_read_proc},
>> - {"locks", locks_read_proc},
>> {"execdomains", execdomains_read_proc},
>> {NULL,}
>> };
>> @@ -934,6 +935,7 @@ void __init proc_misc_init(void)
>> entry->proc_fops = &proc_kmsg_operations;
>> }
>> #endif
>> + create_seq_entry("locks", 0, &proc_locks_operations);
>> create_seq_entry("devices", 0, &proc_devinfo_operations);
>> create_seq_entry("cpuinfo", 0, &proc_cpuinfo_operations);
>> #ifdef CONFIG_BLOCK
>

```
