

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken)

## [PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken)

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-08/msg08432.html>

---

- *From:* Denys Vlasenko <[vda.linux@xxxxxxxxxxxxxxxxx](mailto:vda.linux@xxxxxxxxxxxxxxxxx)>
  - *Date:* Sun, 19 Aug 2007 13:50:21 +0100
- 

On Tuesday 14 August 2007 13:33, Alan Cox wrote:

b) Make recv(fd, buf, size, flags) and send(fd, buf, size, flags); work with non-socket fds too, for flags==0 or flags==MSG\_DONTWAIT. (it's ok to fail with "socket op on non-socket fd" for other values of flags)

I think that makes a lot of sense, and to be honest other MSG\_ flags make useful sense and have meaningful semantics that might be helpful elsewhere if ever coded that way.

Yes, that's my feeling too.

If you want to do this the first job is going to be to sort out the way non-block is propogated to device driver read/write handlers. At the moment they all check filp->f\_flags

...which happens in ~250 files. I'd rather not touch that much of code, if possible.

Attached patch detects send/recv(fd, buf, size, MSG\_DONTWAIT) on non-sockets and turns them into non-blocking write/read. Since filp->f\_flags appear to be read and modified without any locking, I cannot modify it without potentially affecting other processes accessing the same file through shared struct file.

Therefore I simply make a temporary copy of struct file, set O\_NONBLOCK in it and pass it to vfs\_read/write. Is this heresy? ;) I see only one spinlock in struct file:

```
#ifdef CONFIG_EPOLL
spinlock_t f_ep_lock;
#endif /* #ifdef CONFIG_EPOLL */
```

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken) 1

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken)

Do I need to take it?

Also attached is ndelaytest.c which can be used to test that send(MSG\_DONTWAIT) indeed is failing with EAGAIN if write would block and that other processes never see O\_NONBLOCK set.

Comments?

--

vda

--- linux-2.6.22-rc6.src/fs/read\_write.c Fri Jun 15 19:30:05 2007

+++ linux-2.6.22-rc6\_ndelay/fs/read\_write.c Sun Aug 19 10:43:24 2007

@@ -15,6 +15,7 @@

#include <linux/module.h>

#include <linux/syscalls.h>

#include <linux/pagemap.h>

+#include <linux/socket.h>

#include "read\_write.h"

#include <asm/uaccess.h>

@@ -351,6 +352,36 @@

static inline void file\_pos\_write(struct file \*file, loff\_t pos)

{

file->f\_pos = pos;

+

+

+/+ Helper for send/recv on non-sockets \*/

+ssize\_t rw\_with\_flags(struct file \*file, int fput\_needed, void \_\_user \*buf, size\_t count, unsigned flags)

+{

+ int err;

+ loff\_t pos;

+ struct file \*file\_copy;

+

+ file\_copy = file;

+ if (flags & MSG\_DONTWAIT) {

+ /\* We make copy even if O\_NONBLOCK is already set. \*/

+ /\* We don't want it to change under our feet. \*/

+ file\_copy = kmalloc(sizeof(\*file\_copy), GFP\_KERNEL);

+ memcpy(file\_copy, file, sizeof(\*file\_copy));

+ file\_copy->f\_flags |= O\_NONBLOCK;

+ }

+

+ pos = file\_pos\_read(file);

+ if (flags & MSG\_OOB) /\* MSG\_OOB is reused to mean 'write' \*/

+ err = vfs\_write(file\_copy, buf, count, &pos);

+ else

+ err = vfs\_read(file\_copy, buf, count, &pos);

+ file\_pos\_write(file, pos);

+

+ if (flags & MSG\_DONTWAIT) {

+ kfree(file\_copy);

+ }

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken) 2

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken)

```
+ fput_light(file, fput_needed);
+ return err;
}
```

```
asmlinkage ssize_t sys_read(unsigned int fd, char __user * buf, size_t count)
```

```
--- linux-2.6.22-rc6.src/include/linux/fs.h Wed Jun 27 21:24:18 2007
```

```
+++ linux-2.6.22-rc6_ndelay/include/linux/fs.h Sun Aug 19 10:32:20 2007
```

```
@@ -1154,6 +1154,9 @@
```

```
extern ssize_t vfs_writew(struct file *, const struct iovec __user *,
unsigned long, loff_t *);
```

```
+extern ssize_t rw_with_flags(struct file *, int, void __user *, size_t,
+ unsigned);
```

```
+
```

```
/*
```

```
* NOTE: write_inode, delete_inode, clear_inode, put_inode can be called
* without the big kernel lock held in all filesystems.
```

```
--- linux-2.6.22-rc6.src/net/socket.c Fri Jun 15 19:30:08 2007
```

```
+++ linux-2.6.22-rc6_ndelay/net/socket.c Sun Aug 19 11:34:07 2007
```

```
@@ -1585,8 +1585,17 @@
```

```
goto out;
```

```
sock = sock_from_file(sock_file, &err);
```

```
- if (!sock)
```

```
- goto out_put;
```

```
+ if (!sock) {
```

```
+ if (addr)
```

```
+ goto out_put;
```

```
+ if (flags & ~MSG_DONTWAIT)
```

```
+ goto out_put;
```

```
+ /* it's not a socket, but we support a special case:
```

```
+ * send(fd, buf, count, MSG_DONTWAIT)
```

```
+ * (MSG_OOB is reused to mean 'write') */
```

```
+ return rw_with_flags(sock_file, fput_needed, buff, len, flags | MSG_OOB);
```

```
+ }
```

```
+
```

```
iov.iov_base = buff;
```

```
iov.iov_len = len;
```

```
msg.msg_name = NULL;
```

```
@@ -1646,8 +1655,15 @@
```

```
goto out;
```

```
sock = sock_from_file(sock_file, &err);
```

```
- if (!sock)
```

```
- goto out_put;
```

```
+ if (!sock) {
```

```
+ if (addr)
```

```
+ goto out_put;
```

```
+ if (flags & ~MSG_DONTWAIT)
```

```
+ goto out_put;
```

```
+ /* it's not a socket, but we support a special case:
```

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken) 3

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken)

```
+ * recv(fd, ubuf, size, MSG_DONTWAIT) */
+ return rw_with_flags(sock_file, fput_needed, ubuf, size, flags);
+ }

msg.msg_control = NULL;
msg.msg_controllen = 0;
#include <sys/types.h>
#include <sys/socket.h>
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <time.h>
#include <signal.h>

#define SECONDS 10

#define STR "."
// #define STR "123456789 123456789 123456789 123456789 "

/* To see send() resulting in EAGAIN:
 * strace -ff -o log ndelaytest | while sleep 11; do break; done
 * log.$PID:
 * send(1, "123456789 123456789 123456789 12"..., 40, MSG_DONTWAIT)
 * = -1 EAGAIN (Resource temporarily unavailable)
 */

int main()
{
    pid_t pid;
    time_t t;
    int fl;

    puts("starting");
    t = time(0);

    pid = fork();
    if (pid == 0) {
        /* child */
        while ((time(0) - t) < SECONDS-1) {
            #if 0
            /* Uncomment this part and simply run the executable
            * to see race detection code in action */
            #define OP "write"
            fcntl(1, F_SETFL, fcntl(1, F_GETFL) | O_NONBLOCK);
            fl = write(1, STR, sizeof(STR) - 1);
            fcntl(1, F_SETFL, fcntl(1, F_GETFL) & ~O_NONBLOCK);
            #else
            /* This part tests whether send(MSG_DONTWAIT)
            * is racy or not */
            #define OP "send"
```

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken) 4

[PATCH] allow send/recv(MSG\_DONTWAIT) on non-sockets (was Re: O\_NONBLOCK is broken)

```
fl = send(1, STR, sizeof(STR) - 1, MSG_DONTWAIT);
#endif
if (fl < 0) {
perror(OP);
kill(getppid(), SIGKILL);
return 1;
}
}
return 0;
}
```

```
while ((time(0) - t) < SECONDS) {
fl = fcntl(1, F_GETFL);
if (fl & O_NONBLOCK) {
fprintf(stderr, "NONBLOCK:1\n");
kill(pid, SIGKILL);
fcntl(1, F_SETFL, fl & ~O_NONBLOCK);
return 1;
}
}
fprintf(stderr, "NONBLOCK:0\n");
return 0;
}
```