

## Re: What's up with our stdout?

---

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2006-06/msg00097.html>

---

- *From:* Bruce Evans <[bde@xxxxxxxxxxxx](mailto:bde@xxxxxxxxxxxx)>
  - *Date:* Mon, 26 Jun 2006 19:31:10 +1000 (EST)
- 

On Mon, 26 Jun 2006, Andrew Reilly wrote:

On Mon, Jun 26, 2006 at 01:10:38AM +1000, Bruce Evans wrote:

This doesn't seem to have anything to do with stdout. F\_SETLKW just seems to be broken on all regular files (and thus is unsupported for all file types). The above works under the modified version of FreeBSD-5.2 that I use, but it fails with the documented errno EOPNOTSUPP under at least FreeBSD-6.0-STABLE. Replacing STDOUT\_FILENO by fd = open("foo", O\_RDWR) gives the same failure. Replacing FSETLKW by FSETLK or F\_GETLK gives the same failure.

Thanks for the clarification.

Don't all of the databases rely onfcntl locks? How can this be broken?

The problem seems to be that the file system really doesn't support locking. On freefall, both fcntl(2) locking and flock(2) fail in my home directory but work in /tmp. My home directory on freefall is nfs-mounted without nolockd, and also without rpc.lockd or rpc.statd. nfs without the rpc daemons really doesn't support remote locking, so it is correct for it to fail, but rpc.lockd is buggy so it is often not used. On my own machines I normally avoid nfs-locking using nolockd (this gives locking that doesn't work (remotely) but claims to work). On the FreeBSD cluster nfs-locking is apparently normally avoided by nfs-mounting without nolockd and not starting the rpc daemons (this gives locking that doesn't work and doesn't claim to work).

Configuring of locking for nfs is confusing and poorly documented. Neither rpc.lockd nor rpc.statd gets started automatically when a file system is nfs-mounted without nolockd. This wouldn't be easy to automate, since the daemons must be started on both the clients and servers. mount\_nfs(8) doesn't say clearly which daemons must be started

## Re: What's up with our stdout?

where. rc.conf(5) says wrongly that rpc\_lock\_lockd and rpc\_statd\_enable only apply to servers. Starting them both on clients and servers seems to be needed. With a filesystem nfs-remounted without nolockd: there seem to be ordering or timing requirements for starting them -- starting them manually sometimes gave a useful error message for flock() attempts when not all were started, but sometimes starting them all didn't stop flock() from failing and other times gave a hung flock(). Killing and restarting rpc.lockd on the client (while leaving the other daemons running) usually worked to unhang flock() and make it work on the next try.

A modified version of the NetBSD code to test both flock() and fcntl() locking:

```
%%%
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

struct flock stdout_lock;

main()
{
if (flock(STDOUT_FILENO, LOCK_EX) == -1)
err(EXIT_FAILURE, "flock(...LOCK_EX): stdout");
warnx("flock(...LOCK_EX): succeeded");
if (flock(STDOUT_FILENO, LOCK_UN) == -1)
err(EXIT_FAILURE, "flock(...LOCK_UN): stdout");
warnx("flock(...LOCK_EX): succeeded");
stdout_lock.l_len = 0;
stdout_lock.l_start = 0;
stdout_lock.l_type = F_WRLCK;
stdout_lock.l_whence = SEEK_SET;
if (fcntl(STDOUT_FILENO, F_SETLKW, &stdout_lock) == -1)
err(EXIT_FAILURE, "fcntl(...F_SETLKW): stdout");
warnx("fcntl(...F_SETLKW): succeeded");
return (0);
}
%%%
```

The Minix regression tests showed too many other regressions. One was that after mkdir()/open()/rmdir() of a directory, fstat() on the open unlinked directory gave a wrong link count of 2. Another was that after creation of a directory with LINK\_MAX links, it was possible to mkdir() another subdir in the directory, giving 1 more link than the maximum possible:

```
drwxrwxrwx 32768 bde bde 1048576 Jun 25 16:42 DIR_28/foo/
```

This is a more serious bug, since ffs uses the test (i\_nlink <= 0) in a couple of places, and since i\_nlink\_t is int16\_t, 32768 for a link

Re: What's up with our stdout?

Re: What's up with our stdout?

count is unrepresentable (it overflows to -32768). This seems to be caused by either a race in soft updates or using `i_nlink` where `i_effnlink` should be used. These bugs show up on an nfs-mounted directory machines in the FreeBSD cluster. I think nfs doesn't affect this, and the underlying file system is ffs2 with soft updates. Normally I don't want to see bugs like this, and I use ffs1 without soft updates on my own machines to avoid seeing new ones.

Bruce

---

freebsd-arch@xxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@xxxxxxxxxxx"