NAME

lockf - execute a command while holding a file lock

SYNOPSIS

lockf [**-knsw**] [**-t** *seconds*] *file command* [*arguments*]
lockf [**-s**] [**-t** *seconds*] *fd*

DESCRIPTION

The **lockf** utility acquires an exclusive lock on a *file*, creating it if necessary, *and removing the file on exit unless explicitly told not to.* While holding the lock, it executes a *command* with optional *arguments*. After the *command* completes, **lockf** releases the lock, and removes the *file* unless the **-k** option is specified. BSD-style locking is used, as described in flock(2); the mere existence of the *file* is not considered to constitute a lock.

lockf may also be used to operate on a file descriptor instead of a file. If no *command* is supplied, then *fd* must be a file descriptor. The version with a *command* may also be used with a file descriptor by supplying it as a path */dev/fd/N*, where N is the desired file descriptor. The **-k** option is implied when a file descriptor is in use, and the **-n** and **-w** options are silently ignored. This can be used to lock inside a shell script.

If the **lockf** utility is being used to facilitate concurrency between a number of processes, it is recommended that the **-k** option be used. This will guarantee lock ordering, as well as implement a performance enhanced algorithm which minimizes CPU load associated with concurrent unlink, drop and re-acquire activity. It should be noted that if the **-k** option is not used, then no guarantees around lock ordering can be made.

The following options are supported:

-k Causes the lock file to be kept (not removed) after the command completes.

-s Causes **lockf** to operate silently. Failure to acquire the lock is indicated only in the exit status.

-n Causes **lockf** to fail if the specified lock *file* does not exist. If **-n** is not specified, **lockf** will create *file* if necessary.

-t *seconds* Specifies a timeout for waiting for the lock. By default, **lockf** waits indefinitely to acquire the lock. If a timeout is specified with this option, **lockf** will wait at most the given number of *seconds* before giving up. A timeout of 0 may be given, in which case **lockf** will fail unless it can acquire the lock immediately. When a lock times out, *command* is

not executed.

-w Causes **lockf** to open *file* for writing rather than reading. This is necessary on filesystems (including NFSv4) where a file which has been opened read-only cannot be exclusively locked.

In no event will **lockf** break a lock that is held by another process.

EXIT STATUS

If **lockf** successfully acquires the lock, it returns the exit status produced by *command*. Otherwise, it returns one of the exit codes defined in sysexits(3), as follows:

EX_TEMPFAIL The specified lock file was already locked by another process.

EX_CANTCREAT The **lockf** utility was unable to create the lock file, e.g., because of insufficient access privileges.

EX_UNAVAILABLE
The **-n** option is specified and the specified lock file does not exist.

EX_USAGE There was an error on the **lockf** command line.

EX_OSERR A system call (e.g., fork(2)) failed unexpectedly.

EX_SOFTWARE The *command* did not exit normally, but may have been signaled or stopped.

EXAMPLES

The first job takes a lock and sleeps for 5 seconds in the background. The second job tries to get the lock and timeouts after 1 second (PID numbers will differ):

```
$ lockf mylock sleep 5 & lockf -t 1 mylock echo "Success"
[1] 94410
lockf: mylock: already locked
```

The first job takes a lock and sleeps for 1 second in the background. The second job waits up to 5 seconds to take the lock and echoes the message on success (PID numbers will differ):

```
$ lockf mylock sleep 1 & lockf -t 5 mylock echo "Success"
[1] 19995
Success
```

```
[1]+  Done                    lockf mylock sleep 1
```

Lock a file and run a script, return immediately if the lock is not available. Do not delete the file afterward so lock order is guaranteed.

```
$ lockf -t 0 -k /tmp/my.lock myscript
```

Protect a section of a shell script with a lock, wait up to 5 seconds for it to become available. Note that the shell script has opened the lock file */tmp/my.lock*, and **lockf** is performing the lock call exclusively via the passed in file descriptor (9). In this case **-k** is implied, and **-w** has no effect because the file has already been opened by the shell. This example assumes that '>' is implemented in the shell by opening and truncating */tmp/my.lock*, rather than by replacing the lock file.

```
(
        lockf -s -t 5 9
        if [ $? -ne 0 ]; then
                echo "Failed to obtain lock"
                exit 1
        fi

        echo Start
        # Do some stuff
        echo End
) 9>/tmp/my.lock
```

SEE ALSO

flock(2), lockf(3), sysexits(3)

HISTORY

A **lockf** utility first appeared in FreeBSD 2.2.

AUTHORS

John Polstra <*jdp@polstra.com*>