NAME

futex − Fast Userspace Locking system call

SYNOPSIS

```
#include <linux/futex.h>
#include <sys/time.h>

int futex(int *uaddr, int op, int val, const struct timespec *timeout,
          int *uaddr2, int val3);
```

DESCRIPTION

The **futex**() system call provides a method for a program to wait for a value at a given address to change, and a method to wake up anyone waiting on a particular address (while the addresses for the same memory in separate processes may not be equal, the kernel maps them internally so the same memory mapped in different locations will correspond for **futex**() calls). It is typically used to implement the contended case of a lock in shared memory, as described in **futex**(7).

When a **futex**(7) operation did not finish uncontended in userspace, a call needs to be made to the kernel to arbitrate. Arbitration can either mean putting the calling process to sleep or, conversely, waking a waiting process.

Callers of this function are expected to adhere to the semantics as set out in **futex**(7). As these semantics involve writing non-portable assembly instructions, this in turn probably means that most users will in fact be library authors and not general application developers.

The *uaddr* argument needs to point to an aligned integer which stores the counter. The operation to execute is passed via the *op* argument, along with a value *val*.

Five operations are currently defined:

FUTEX_WAIT

This operation atomically verifies that the futex address *uaddr* still contains the value *val*, and sleeps awaiting **FUTEX_WAKE** on this futex address. If the *timeout* argument is non-NULL, its contents describe the maximum duration of the wait, which is infinite otherwise. The arguments *uaddr2* and *val3* are ignored.

For **futex**(7), this call is executed if decrementing the count gave a negative value (indicating contention), and will sleep until another process releases the futex and executes the **FUTEX_WAKE** operation.

FUTEX_WAKE

This operation wakes at most *val* processes waiting on this futex address (i.e., inside **FUTEX_WAIT**). The arguments *timeout*, *uaddr2* and *val3* are ignored.

For **futex**(7), this is executed if incrementing the count showed that there were waiters, once the futex value has been set to 1 (indicating that it is available).

FUTEX_FD (present up to and including Linux 2.6.25)

To support asynchronous wakeups, this operation associates a file descriptor with a futex. If another process executes a **FUTEX_WAKE**, the process will receive the signal number that was passed in *val*. The calling process must close the returned file descriptor after use. The arguments *timeout*, *uaddr2* and *val3* are ignored.

To prevent race conditions, the caller should test if the futex has been upped after **FUTEX_FD** returns.

Because it was inherently racy, **FUTEX_FD** has been removed from Linux 2.6.26 onwards.

FUTEX_REQUEUE (since Linux 2.5.70)
This operation was introduced in order to avoid a "thundering herd" effect when **FUTEX_WAKE** is used and all processes woken up need to acquire another futex. This call wakes up *val* processes, and requeues all other waiters on the futex at address *uaddr2*. The arguments *timeout* and *val3* are ignored.

FUTEX_CMP_REQUEUE (since Linux 2.6.7)
There was a race in the intended use of **FUTEX_REQUEUE**, so **FUTEX_CMP_REQUEUE** was introduced. This is similar to **FUTEX_REQUEUE**, but first checks whether the location *uaddr* still contains the value *val3*. If not, the operation fails with the error **EAGAIN**. The argument *timeout* is ignored.

RETURN VALUE

Depending on which operation was executed, the returned value for a successful call can have differing meanings.

FUTEX_WAIT
Returns 0 if the process was woken by a **FUTEX_WAKE** call. In case of timeout, the operation fails with the error **ETIMEDOUT**. If the futex was not equal to the expected value, the operation fails with the error **EWOULDBLOCK**. Signals (see **signal**(7)) or other spurious wakeups cause **FUTEX_WAIT** to fail with the error **EINTR**.

FUTEX_WAKE
Returns the number of processes woken up.

FUTEX_FD
Returns the new file descriptor associated with the futex.

FUTEX_REQUEUE
Returns the number of processes woken up.

FUTEX_CMP_REQUEUE
Returns the number of processes woken up.

In the event of an error, all operations return −1, and set *errno* to indicate the error.

ERRORS

EACCES
No read access to futex memory.

EAGAIN
FUTEX_CMP_REQUEUE found an unexpected futex value. (This probably indicates a race; use the safe **FUTEX_WAKE** now.)

EFAULT
Error in getting *timeout* information from userspace.

EINVAL
An operation was not defined or error in page alignment.

ENFILE
The system limit on the total number of open files has been reached.

ENOSYS
Invalid operation specified in *op*.

VERSIONS

Initial futex support was merged in Linux 2.5.7 but with different semantics from what was described above. A 4-argument system call with the semantics given here was introduced in Linux 2.5.40. In Linux 2.5.70 one argument was added. In Linux 2.6.7 a sixth argument was added — messy, especially on the s390 architecture.

CONFORMING TO

This system call is Linux-specific.

NOTES

To reiterate, bare futexes are not intended as an easy-to-use abstraction for end-users. (There is no wrapper function for this system call in glibc.) Implementors are expected to be assembly literate and to have read the sources of the futex userspace library referenced below.

SEE ALSO

futex(7)

Fuss, Futexes and Furwocks: Fast Userlevel Locking in Linux (proceedings of the Ottawa Linux Symposium 2002), futex example library, futex-*.tar.bz2 <URL:ftp://ftp.nl.kernel.org/pub/linux/kernel/people/rusty/>.

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.