futex2

Author: AndrÃ© Almeida <andrealmeid@collabora.com>

futex, or fast user mutex, is a set of syscalls to allow userspace to create performant synchronization mechanisms, such as mutexes, semaphores and conditional variables in userspace. C standard libraries, like glibc, uses it as a means to implement more high level interfaces like pthreads.

futex2 is a followup version of the initial futex syscall, designed to overcome limitations of the original interface.

User API

`futex_waitv()`

Wait on an array of futexes, wake on any:

```
futex_waitv(struct futex_waitv *waiters, unsigned int nr_futexes,
            unsigned int flags, struct timespec *timeout, clockid_t clockid)

struct futex_waitv {
      __u64 val;
      __u64 uaddr;
      __u32 flags;
      __u32 __reserved;
};
```

Userspace sets an array of struct futex_waitv (up to a max of 128 entries), using `uaddr` for the address to wait for, `val` for the expected value and `flags` to specify the type (e.g. private) and size of futex. `__reserved` needs to be 0, but it can be used for future extension. The pointer for the first item of the array is passed as `waiters`. An invalid address for `waiters` or for any `uaddr` returns `-EFAULT`.

If userspace has 32-bit pointers, it should do a explicit cast to make sure the upper bits are zeroed. `uintptr_t` does the tricky and it works for both 32/64-bit pointers.

`nr_futexes` specifies the size of the array. Numbers out of [1, 128] interval will make the syscall return `-EINVAL`.

The `flags` argument of the syscall needs to be 0, but it can be used for future extension.

For each entry in `waiters` array, the current value at `uaddr` is compared to `val`. If it's different, the syscall undo all the work done so far and return `-EAGAIN`. If all tests and verifications succeeds, syscall waits until one of the following happens:

- The timeout expires, returning `-ETIMEOUT`.
- A signal was sent to the sleeping task, returning `-ERESTARTSYS`.
- Some futex at the list was woken, returning the index of some waked futex.

An example of how to use the interface can be found at `tools/testing/selftests/futex/functional/futex_waitv.c`.

Timeout

`struct timespec *timeout` argument is an optional argument that points to an absolute timeout. You need to specify the type of clock being used at `clockid` argument. `CLOCK_MONOTONIC` and `CLOCK_REALTIME` are supported. This syscall accepts only 64bit timespec structs.

Types of futex

A futex can be either private or shared. Private is used for processes that shares the same memory space and the virtual address of the futex will be the same for all processes. This allows for optimizations in the kernel. To use private futexes, it's necessary to specify `FUTEX_PRIVATE_FLAG` in the futex flag. For processes that doesn't share the same memory space and therefore can have different virtual addresses for the same futex (using, for instance, a file-backed shared memory) requires different internal mechanisms to be get properly enqueued. This is the default behavior, and it works with both private and shared futexes.

Futexes can be of different sizes: 8, 16, 32 or 64 bits. Currently, the only supported one is 32 bit sized futex, and it need to be specified using `FUTEX_32` flag.