futex描述

futex()系统调用提供了一种方法用于等待某个特定条件的发生。一种典型的应用是在共享内存同步中作为阻塞装置。当使用futex时，绝大部分同步操作都在用户态完成，一个用户态程序只有在有可能阻塞一段较长的时间等待条件的发生时才使用futex。其他的futex操作可以用来唤醒等待特定条件发生的任何进程或线程。

一个futex是一个32位的值，它的地址通过futex()传入，futexs在所有的平台上都是32位的(包括64位系统上)，所有的futex操作都是在这个值上。为了在多个进程间共享futex，futex位于由mmap或者shmat创建的一块共享内存上(这种情况下，futex值在不同的进程上可能位于不同的虚拟地址，但这些虚拟地址都指向相同的物理地址)。在多线程程序中，将futex值放到一个所有线程共享的全局变量中就可以了。

当执行一个futex操作请求阻塞一个线程时，只有在 *uaddr == val 时，内核才会执行阻塞操作，这个操作中的所有步骤(1. 导入*uaddr的值; 2. 比较; 3.阻塞线程)将原子的执行，并且当其他线程在同一个futex值上并行操作时所有的步骤不会乱序。

futex的一种使用方式是用来实现锁，锁的状态(acquired or not acquired)可以用在共享内存中的原子标志表示。在非竞争的情况下，线程可以通过原子操作访问和修改锁的状态(这些操作全部在用户态操作，内核不会保存任何关于锁的状态)。从另一方面来说，另一个线程可能无法获取锁(因为锁已经被某个线程获取)，这个线程将通过如下这种方式执行futex()等待操作：

```
atomic<int> lock; // lock :  0. 锁未被获取 1.锁被获取
futex(&lock, FUTEX_WAIT, 1, NULL, NULL, NULL);
```

futex(FUTEX_WAIT)将会检测lock的值，只有在等于1时才阻塞线程。当线程释放锁时，该线程必须首先重置锁的状态，然后执行futex操作唤醒阻塞在lock上的线程。

注意使用futex并没有显示的初始化和销毁操作，内核只有在一个指定的futex值上执行futex操作时(例如FUTEX_WAIT)才会维护futex数据。

参数

uaddr指向futex值，在所有的平台上，futex值是一个4字节的整数并且必须4字节对齐。

对于某些阻塞操作，timeout参数是一个指向timespec结构的指针，表明了操作的超时时间。然而，在其他的某些操作下，它的最低4字节被作为一个整数值，这个整数值的含义因futex操作的不同而不同，对于这些操作来说，内核会将timeout值转换为unsigned long，然后转换为uint32_t，在接下来的说明中，它将表示为val2。

在需要的时候，uaddr2是指向第二个futex值的指针，val3的解释将依赖于具体的操作

操作

FUTEX_CLOCK_REALTIME (since Linux 2.6.28)

这个标志选项只能用在**FUTEX_WAIT_BITSET** 和 **FUTEX_WAIT_REQUEUE_PI** 操作中，如果设置了这个标志选项，内核会将timeout视作基于**CLOCK_REALTIME** 的绝对时间。如果没有设置这个标志，则内核将它视作基于**CLOCK_MONOTONIC**的相对时间。

FUTEX_WAIT (since Linux 2.6.0)

*本操作会检查uaddr是否等于val，如果相等的话，则睡眠等待在uaddr上的**FUTEX_WAKE** 操作，如果线程开始睡眠，则它被认为是这个futex值上一个等待者。如果两者不相等，则操作失败并返回 error EAGAIN。比较*uaddr和val的目的是为了避免丢失唤醒。

如果timeout参数不为NULL，则它指定了等待的超时时间(根据**CLOCK_MONOTONIC测量得出**)。

FUTEX_WAKE (since Linux 2.6.0)

本操作会唤醒最多val个等待者，绝大部分情况下，val的值为1(只唤醒一个等待者)或INT_MAX(唤醒所有的等待者) 。**注意没有任何机制保证一定唤醒某些特定的等待者(比如被认为是高优先级的等待者)**

参数timeout, uaddr2, val3将被忽略。

代码

```
fn futex_wait(
    vaddr: VirtAddr,
    futex_op: i32,
    val: u32,
    timeout: usize,
) -> Result<usize, SyscallError> {
    let mut to = false;
    let current_task = current_task();
    let deadline = if timeout != 0 {
        Some(Duration::from_nanos(timeout as u64) + axhal::time::current_time())
    } else {
        None
    };
    loop {
        let key = get_futex_key(vaddr, futex_op);
        let process = current_process();
        if process.manual_alloc_for_lazy(vaddr).is_ok() {
            let real_futex_val = unsafe { (vaddr.as_usize() as *const
u32).read_volatile() };
            info!("real val: {:#x}, expected val: {:#x}", real_futex_val, val);
            if real_futex_val != val {
                return Err(SyscallError::EAGAIN);
            }
            futex_quque(key, &current_task, val);

            if let Some(deadline) = deadline {
                let now = axhal::time::current_time();
                to = deadline < now;
            }
            if timeout == 0 || !to {
                yield_now_task();
            }
            // If we were woken (and unqueued), we succeeded, whatever.
            // TODO: plist_del, not just iterate all the list
            if !futex_unqueue(key, &current_task) {
```

```
                return Ok(0);
            }
            if to {
                return Err(SyscallError::ETIMEDOUT);
            }
            // we expect signal_pending(current), but we might be the victim
            // of a spurious wakeup as well.
            #[cfg(feature = "signal")]
            if process.have_signals().is_some() {
                // 被信号打断
                return Err(SyscallError::EINTR);
            }
        } else {
            return Err(SyscallError::EFAULT);
        }
    }
}

fn futex_wait_bitset(
    vaddr: VirtAddr,
    futex_op: i32,
    val: u32,
    timeout: usize,
    bitset: u32,
) -> Result<usize, SyscallError> {
    let mut to = false;
    let current_task = current_task();
    let deadline = if timeout != 0 {
        Some(Duration::from_nanos(timeout as u64) + axhal::time::current_time())
    } else {
        None
    };
    loop {
        let key = get_futex_key(vaddr, futex_op);
        let process = current_process();
        if process.manual_alloc_for_lazy(vaddr).is_ok() {
            let real_futex_val = unsafe { (vaddr.as_usize() as *const
u32).read_volatile() };
            info!("real val: {:#x}, expected val: {:#x}", real_futex_val, val);
            if real_futex_val != val {
                return Err(SyscallError::EAGAIN);
            }
            futex_quque(key, &current_task, val);

            // Check the bitset condition
            if (real_futex_val & bitset) == bitset {
                return Ok(0);
            }

            if let Some(deadline) = deadline {
                let now = axhal::time::current_time();
                to = deadline < now;
            }
            if timeout == 0 || !to {
                yield_now_task();
```

```
            }
            // If we were woken (and unqueued), we succeeded, whatever.
            // TODO: plist_del, not just iterate all the list
            if !futex_unqueue(key, &current_task) {
                return Ok(0);
            }
            if to {
                return Err(SyscallError::ETIMEDOUT);
            }
            // we expect signal_pending(current), but we might be the victim
            // of a spurious wakeup as well.
            #[cfg(feature = "signal")]
            if process.have_signals().is_some() {
                // 被信号打断
                return Err(SyscallError::EINTR);
            }
        } else {
            return Err(SyscallError::EFAULT);
        }
    }
}

fn futex_wake(vaddr: VirtAddr, futex_op: i32, val: u32) -> Result<usize,
SyscallError> {
    let mut ret = 0;
    let key = get_futex_key(vaddr, futex_op);
    // 当前任务释放了锁，所以不需要再次释放
    let mut futex_wait_task = FUTEX_WAIT_TASK.lock();
    if futex_wait_task.contains_key(&key) {
        let wait_list = futex_wait_task.get_mut(&key).unwrap();
        // info!("now task: {}", wait_list.len());
        while let Some((task, _)) = wait_list.pop_front() {
            // 唤醒一个正在等待的任务
            info!("wake task: {}", task.id().as_u64());
            WAIT_FOR_FUTEX.notify_task(&task);
            ret += 1;
            if ret == val {
                break;
            }
        }
    }
    drop(futex_wait_task);
    yield_now_task();
    Ok(ret as usize)
}

fn futex_wake_bitset(
    vaddr: VirtAddr,
    futex_op: i32,
    val: u32,
    bitset: u32,
) -> Result<usize, SyscallError> {
    let mut ret = 0;
    let key = get_futex_key(vaddr, futex_op);
    let mut futex_wait_task = FUTEX_WAIT_TASK.lock();
```

```
    if let Some(wait_list) = futex_wait_task.get_mut(&key) {
        for (task, task_bitset) in wait_list.iter() {
            let wakeup = (val == 0 || ret < val) && ((*task_bitset & bitset) !=
0);
            if wakeup {
                WAIT_FOR_FUTEX.notify_task(&task);
                ret += 1;
            }
            if val != 0 && ret >= val {
                break;
            }
        }

        wait_list.retain(|(_, task_bitset)| (*task_bitset & !bitset) != 0);
        if wait_list.is_empty() {
            futex_wait_task.remove(&key);
        }
    }
    drop(futex_wait_task);
    Ok(ret as usize)
}

/// To do the futex operation
///
/// It may create, remove the futex wait task or requeue the futex wait task
pub fn futex(
    vaddr: VirtAddr,
    futex_op: i32,
    val: u32,
    timeout: usize,
    vaddr2: VirtAddr,
    val2: usize,
    val3: u32,
) -> Result<usize, SyscallError> {
    let flag = FutexFlags::new(futex_op);
    match flag {
        FutexFlags::Wait => futex_wait(vaddr, futex_op, val, timeout),
        FutexFlags::Wake => futex_wake(vaddr, futex_op, val),
        FutexFlags::Requeue => {
            futex_requeue(val, val2, vaddr, vaddr2);
            Ok(0)
        }
        FutexFlags::WaitBitset | FutexFlags::RealTime => {
            futex_wait_bitset(vaddr, futex_op, val, timeout, val3)
        }
        FutexFlags::WakeBitset => futex_wake_bitset(vaddr, futex_op, val, val3),
        _ => Err(SyscallError::EINVAL),
    }
}

/// # Arguments
/// * vaddr: usize
/// * futex_op: i32
/// * futex_val: u32
/// * time_out_val: usize
```

```
/// * vaddr2: usize
/// * val3: u32
pub fn syscall_futex(args: [usize; 6]) -> SyscallResult {
    let vaddr = args[0];
    let futex_op = args[1] as i32;
    let futex_val = args[2] as u32;
    let time_out_val = args[3];
    let vaddr2 = args[4];
    let val3 = args[5] as u32;
    let process = current_process();
    let timeout = if time_out_val != 0 &&
process.manual_alloc_for_lazy(time_out_val.into()).is_ok()
    {
        let time_sepc: TimeSecs = unsafe { *(time_out_val as *const TimeSecs) };
        time_sepc.turn_to_nanos()
    } else {
        // usize::MAX
        0
    };
    // 释放锁，防止任务无法被调度
    match futex(
        vaddr.into(),
        futex_op,
        futex_val,
        timeout,
        vaddr2.into(),
        time_out_val,
        val3,
    ) {
        Ok(ans) => Ok(ans as isize),
        Err(errno) => Err(errno),
    }
}
```

相关截图


mlock.c
futex_wait_timeout.c
syscalls.c 3
futextest.h 2
futex.h ...\futex
linux-master > kernel > futex > syscalls.c > do_futex(u32 __user *, int, u32, ktime_t *, u32 __user *, u32, u32)
80
81 return ret;
82 }
83
84 long do_futex(u32 __user *uaddr, int op, u32 val, ktime_t *timeout,
85 u32 __user *uaddr2, u32 val2, u32 val3)
86 {
87 unsigned int flags = futex_to_flags(op);
88 int cmd = op & FUTEX_CMD_MASK;
89
90 if (flags & FLAGS_CLOCKRT) {
91 if (cmd != FUTEX_WAIT_BITSET &&
92 cmd != FUTEX_WAIT_REQUEUE_PI &&
93 cmd != FUTEX_LOCK_PI2)
94 return -ENOSYS;
95 }
96
97 switch (cmd) {
98 case FUTEX_WAIT:
99 val3 = FUTEX_BITSET_MATCH_ANY;
100 fallthrough;
101 case FUTEX_WAIT_BITSET:
102 return futex_wait(uaddr, flags, val, timeout, val3);
103 case FUTEX_WAKE:
104 val3 = FUTEX_BITSET_MATCH_ANY;
105 fallthrough;
106 case FUTEX_WAKE_BITSET:
107 return futex_wake(uaddr, flags, val, val3);
108 case FUTEX_REQUEUE:
109 return futex_requeue(uaddr, flags, uaddr2, flags, val, val2, NULL, 0);
110 case FUTEX_CMP_REQUEUE:
111 return futex_requeue(uaddr, flags, uaddr2, flags, val, val2, &val3, 0);
112 case FUTEX_WAKE_OP:
113 return futex_wake_op(uaddr, flags, uaddr2, val, val2, val3);
114 case FUTEX_LOCK_PI:
115 flags |= FLAGS_CLOCKRT;
116 fallthrough;
117 case FUTEX_LOCK_PI2:
118 return futex_lock_pi(uaddr, flags, timeout, 0);
119 case FUTEX_UNLOCK_PI:
120 return futex_unlock_pi(uaddr, flags);
121 case FUTEX_TRYLOCK_PI:
122 return futex_lock_pi(uaddr, flags, NULL, 1);
123 case FUTEX_WAIT_REQUEUE_PI:
124 val3 = FUTEX_BITSET_MATCH_ANY;
125 return futex_wait_requeue_pi(uaddr, flags, val, timeout, val3,
126 uaddr2);
127 case FUTEX_CMP_REQUEUE_PI:
128 return futex_requeue(uaddr, flags, uaddr2, flags, val, val2, &val3, 1);
129 }
130 return -ENOSYS;
131 }

apps > c > futex > C futex.c > main()

```c
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/futex.h>

int main() {
    int futex_var = 0;

    // 创建一个 futex，初始值为 0
    int *futex_ptr = &futex_var;

    // 唤醒等待在 futex 上的线程（私有唤醒）
    syscall(SYS_futex, futex_ptr, FUTEX_WAKE_PRIVATE, 1, NULL, NULL, 0);

    return 0;
}
```

```
/*! Time structures.

The `time` module contains structures used to represent bo
absolute and relative time.

 - [Instant] is used to represent absolute time.
 - [Duration] is used to represent relative time.

[Instant]: struct.Instant.html
[Duration]: struct.Duration.html
*/
```

match返回类型必须相同，否则编译器会报错

枚举返回的结果类型，在后面的判断中比较大小时又出现类型不同的问题

```rust
let deadline: () = if timeout != 0 {
    enum MyResult {
        InstantResult(Instant),
        DurationResult(Duration),
        NoneResult,
    }

    let deadline: Option<MyResult> = if timeout != 0 {
        let result: Option<MyResult> = Some(match futex_op {
            FUTEX_CLOCK_REALTIME: i32 => MyResult::InstantResult(Instant::from_micros(timeout as i64) + axhal::time::current_time().into()),
            FUTEX_WAIT_BITSET: i32 => MyResult::DurationResult((axhal::time::current_time() + Duration::from_nanos(timeout as u64)).into()),
            _ => MyResult::NoneResult,
        });
        result
    } else {
    None
    };
};
```

```
root@ubuntu:/home/josen/Starry/testcases/ZLM# strace ./futex
execve("./futex", ["./futex"], 0x7fffb520fa60 /* 25 vars */) = 0
brk(NULL)                               = 0x5639e6655000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd77dffbb0) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=76948, ...}) = 0
mmap(NULL, 76948, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fc143f5c000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300A\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\2C\n\357_\243\335\2449\206V>\237\374\304"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029592, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fc143f5a000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\2C\n\357_\243\335\2449\206V>\237\374\304"..., 68, 880) = 68
mmap(NULL, 2037344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fc143d68000
mmap(0x7fc143d8a000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7fc143d8a000
mmap(0x7fc143f02000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19a000) = 0x7fc143f02000
mmap(0x7fc143f50000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fc143f50000
mmap(0x7fc143f56000, 13920, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fc143f56000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fc143f5b540) = 0
mprotect(0x7fc143f50000, 16384, PROT_READ) = 0
mprotect(0x5639e4e4a000, 4096, PROT_READ) = 0
mprotect(0x7fc143f9c000, 4096, PROT_READ) = 0
munmap(0x7fc143f5c000, 76948)           = 0
futex(0x7ffd77dffb7c, FUTEX_WAKE_PRIVATE, 1) = 0
exit_group(0)                           = ?
```

```
./futex[ 61.045288 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x1ddfb3, len: 1

[ 61.045551 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3ffff7c4, len: 12
        7:     [ 61.045905 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x1ddfb4, len: 1

[ 61.046254 0:8 axstarry::syscall:51] [syscall] id = 20, args = [2, 1073739728, 8, 4294967295, 2236176, 4294967295], return 68
[ 61.046640 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x481e0, fault_vaddr=0x481e0, error_code=0x14
[ 61.047340 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x124870, fault_vaddr=0x124870, error_code=0x14
[ 61.048158 0:8 axstarry::syscall:38] [syscall] id = FUTEX, args = [1073740844, 129, 1, 0, 0, 0], entry
[ 61.048648 0:8 axstarry::syscall:51] [syscall] id = 202, args = [1073740844, 129, 1, 0, 0, 0], return 0
[ 61.049187 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x9e00c, fault_vaddr=0x9e00c, error_code=0x14
[ 61.049854 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x9fa70, fault_vaddr=0x9fa70, error_code=0x14
[ 61.050660 0:8 axstarry::syscall:38] [syscall] id = GETPID, args = [1073739300, 0, 67295889, 67295889, 1073740624, 1], entry
[ 61.051114 0:8 axstarry::syscall:51] [syscall] id = 39, args = [1073739300, 0, 67295889, 67295889, 1073740624, 1], return 7
[ 61.051594 0:8 axstarry::syscall:29] [syscall] id = WRITEV, args = [2, 1073739312, 10, 4294967295, 1073739225, 4294967295], entry
[ 61.052023 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3ffff624, len: 12
        7:     [ 61.052276 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x402da91, len: 1

[ 61.052460 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3ffff624, len: 12
        7:     [ 61.052921 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x402da92, len: 14
calling fini: [ 61.053286 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3ffffe27, len: 7
./futex[ 61.053656 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x402daa2, len: 2
```

```
[276.823925 0:10 axstarry::syscall:38] [syscall] id = FUTEX, args = [1073740844, 137, 1, 0, 0, 18446744073709551615], entry
[276.824276 0:10 axstarry::syscall_task::imp::futex:171] ---------------futex_wait_bitset---------------
[276.824711 0:10 axstarry::syscall_task::imp::futex:180] real val: 0x0, expected val: 0x1
[276.824990 0:10 axstarry::syscall:51] [syscall] id = 202, args = [1073740844, 137, 1, 0, 0, 18446744073709551615], return -11
[276.825296 0:10 axstarry::syscall:38] [syscall] id = FUTEX, args = [1073740844, 138, 1, 0, 0, 18446744073709551615], entry
[276.825618 0:10 axstarry::syscall_task::imp::futex:244] ---------------futex_wake_bitset---------------
[276.825917 0:10 axstarry::syscall:51] [syscall] id = 202, args = [1073740844, 138, 1, 0, 0, 18446744073709551615], return 0
[276.826333 0:10 axhal::arch::x86_64::trap:21] User #PF @ 0x9e00c, fault_vaddr=0x9e00c, error_code=0x14
[276.827185 0:10 axhal::arch::x86_64::trap:21] User #PF @ 0x9fa70, fault_vaddr=0x9fa70, error_code=0x14
[276.827844 0:10 axstarry::syscall:38] [syscall] id = GETPID, args = [1073739300, 0, 67295889, 67295889, 1073740624, 1], entry
[276.828243 0:10 axstarry::syscall:51] [syscall] id = 39, args = [1073739300, 0, 67295889, 67295889, 1073740624, 1], return 9
[276.828667 0:10 axstarry::syscall:29] [syscall] id = WRITEV, args = [2, 1073739312, 10, 4294967295, 1073739225, 4294967295], entry
```

修改后代码为

modules/sxprocess/src/futex.rs:

```
    /// vec中的元素分别是任务指针,对应存储时的futex变量的值

pub static FUTEX_WAIT_BITSET_TASK: Mutex<BTreeMap<FutexKey, VecDeque<(AxTaskRef,
u32, u32)>>> =
Mutex::new(BTreeMap::new());
```

ulib/axstarry/src/syacall_task/imp/futex.rs

```
use axprocess::{
    current_process, current_task,
    futex::{get_futex_key, FutexKey, FutexRobustList, FUTEX_WAIT_TASK,
FUTEX_WAIT_BITSET_TASK, WAIT_FOR_FUTEX},
    yield_now_task,
};
```

```
fn futex_queue_with_bitset(key: FutexKey, curr: &CurrentTask, val: u32, bitset:
u32) {
    let mut futex_wait_bitset_task = FUTEX_WAIT_BITSET_TASK.lock();
    let wait_bitset_list = futex_wait_bitset_task.entry(key).or_default();
    wait_bitset_list.push_back((curr.as_task_ref().clone(), val, bitset));
}



fn futex_unqueue_with_bitset(key: FutexKey, curr: &CurrentTask, bitset: u32) ->
bool {
    let mut futex_wait_bitset_task = FUTEX_WAIT_BITSET_TASK.lock();
    if futex_wait_bitset_task.contains_key(&key) {
        let wait_bitset_list = futex_wait_bitset_task.get_mut(&key).unwrap();
        if let Some(index) = wait_bitset_list
            .iter()
            .position(|(task, _, b)| task.id() == curr.id() && *b == bitset)
        {
            wait_bitset_list.remove(index);
            return true;
        }
    }
    false
}
```

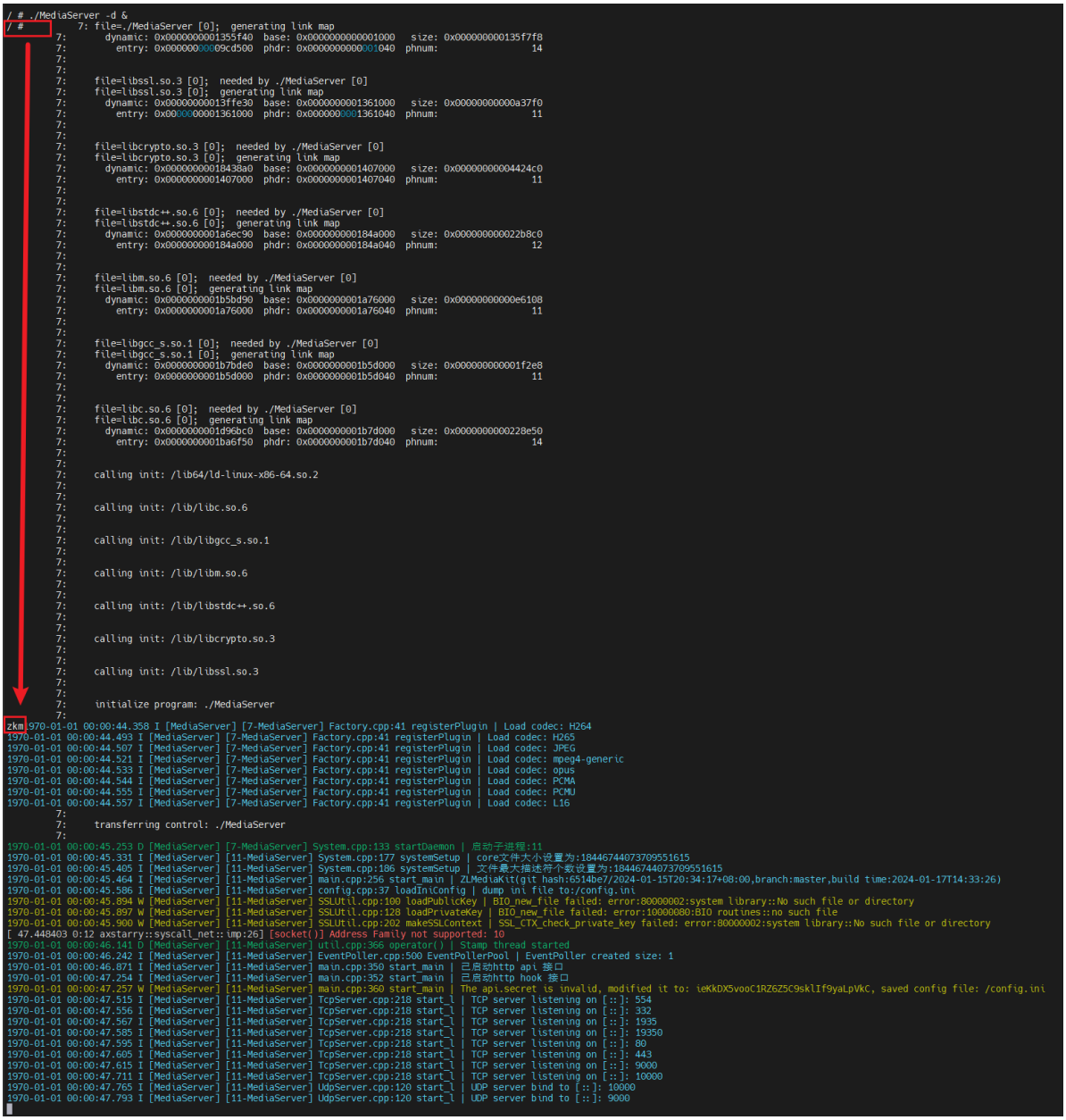
```
fn futex_wait_bitset(
    vaddr: VirtAddr,
    futex_op: i32,
    val: u32,
    timeout: usize,
    bitset: u32,
) -> Result<usize, SyscallError> {
    let mut to = false;
    let current_task = current_task();
    let deadline = if timeout != 0 {
        Some(Duration::from_nanos(timeout as u64) + axhal::time::current_time())
    } else {
        None
    };


    loop {
        let key = get_futex_key(vaddr, futex_op);
        let process = current_process();

        if process.manual_alloc_for_lazy(vaddr).is_ok() {
            let real_futex_val = unsafe { (vaddr.as_usize() as *const
u32).read_volatile() };
            info!("real val: {:#x}, expected val: {:#x}", real_futex_val, val);

            if real_futex_val != val {
                return Err(SyscallError::EAGAIN);
            }
```

```
                futex_queue_with_bitset(key, &current_task, val, bitset);

                if let Some(deadline) = deadline {
                    let now = axhal::time::current_time();
                    to = deadline < now;
                }

                if timeout == 0 || !to {
                    yield_now_task();
                }

                if !futex_unqueue_with_bitset(key, &current_task, bitset) {
                    return Ok(0);
                }

                if to {
                    return Err(SyscallError::ETIMEDOUT);
                }

                #[cfg(feature = "signal")]
                if process.have_signals().is_some() {
                    return Err(SyscallError::EINTR);
                }
            } else {
                return Err(SyscallError::EFAULT);
            }
        }

    }
```

未添加futex_wait_bitset之前执行ZLM的结果为

```
1970-01-01 00:00:14.663 I [MediaServer] [9-MediaServer] Factory.cpp:41 registerPlugin | Load codec: H264
1970-01-01 00:00:14.786 I [MediaServer] [9-MediaServer] Factory.cpp:41 registerPlugin | Load codec: H265
1970-01-01 00:00:14.797 I [MediaServer] [9-MediaServer] Factory.cpp:41 registerPlugin | Load codec: JPEG
1970-01-01 00:00:14.808 I [MediaServer] [9-MediaServer] Factory.cpp:41 registerPlugin | Load codec: mpeg4-generic
1970-01-01 00:00:14.819 I [MediaServer] [9-MediaServer] Factory.cpp:41 registerPlugin | Load codec: opus
1970-01-01 00:00:14.830 I [MediaServer] [9-MediaServer] Factory.cpp:41 registerPlugin | Load codec: PCMA
1970-01-01 00:00:14.841 I [MediaServer] [9-MediaServer] Factory.cpp:41 registerPlugin | Load codec: PCMU
1970-01-01 00:00:14.843 I [MediaServer] [9-MediaServer] Factory.cpp:41 registerPlugin | Load codec: L16
        9:
        9:     transferring control: ./MediaServer
        9:
[ 15.500558 0:10 fatfs::dir:140] Is a directory
[ 15.502442 0:10 fatfs::dir:140] Is a directory
[ 15.504179 0:10 fatfs::dir:140] Is a directory
[ 15.505942 0:10 fatfs::dir:140] Is a directory
[ 15.508128 0:10 fatfs::dir:140] Is a directory
[ 15.510001 0:10 fatfs::dir:140] Is a directory
[ 15.548939 0:10 fatfs::dir:140] Is a directory
1970-01-01 00:00:15.563 D [MediaServer] [9-MediaServer] System.cpp:133 startDaemon | 启动子进程:13
[ 15.589967 0:10 fatfs::dir:140] Is a directory
[ 15.591415 0:10 fatfs::dir:140] Is a directory
1970-01-01 00:00:15.610 I [MediaServer] [13-MediaServer] System.cpp:177 systemSetup | core文件大小设置为:18446744073709551615
[ 15.646279 0:14 fatfs::dir:140] Is a directory
[ 15.647834 0:14 fatfs::dir:140] Is a directory
1970-01-01 00:00:15.664 I [MediaServer] [13-MediaServer] System.cpp:186 systemSetup | 文件最大描述符个数设置为:18446744073709551615
QEMU: Terminated
```

添加之后执行结果为



可以执行进入监听状态，但是目前不能在后台执行命令，按enter后无法继续输入命令，可能存在问题的日志:

```
[ 11.332328 0:8 axstarry::syscall:38] [syscall] id = SIGACTION, args = [22, 1073739808, 0, 8, 11056, 1], entry
[ 11.334617 0:8 axstarry::syscall_task::imp::signal:20] signum: 22, action: 3FFFF820, old_action: 0
[ 11.336530 0:8 axstarry::syscall:51] [syscall] id = 13, args = [22, 1073739808, 0, 8, 11056, 1], return 0
[ 11.338578 0:8 axstarry::syscall:38] [syscall] id = SIGACTION, args = [2, 1073739808, 0, 8, 11056, 1], entry
[ 11.341199 0:8 axstarry::syscall_task::imp::signal:20] signum: 2, action: 3FFFF820, old_action: 0
[ 11.343150 0:8 axstarry::syscall:51] [syscall] id = 13, args = [2, 1073739808, 0, 8, 11056, 1], return 0
[ 11.345361 0:8 axstarry::syscall:38] [syscall] id = SIGACTION, args = [15, 1073739808, 0, 8, 11056, 1], entry
[ 11.347451 0:8 axstarry::syscall_task::imp::signal:20] signum: 15, action: 3FFFF820, old_action: 0
[ 11.349383 0:8 axstarry::syscall:51] [syscall] id = 13, args = [15, 1073739808, 0, 8, 11056, 1], return 0
[ 11.351638 0:8 axstarry::syscall:38] [syscall] id = SIGACTION, args = [3, 1073739808, 0, 8, 11056, 1], entry
[ 11.354755 0:8 axstarry::syscall_task::imp::signal:20] signum: 3, action: 3FFFF820, old_action: 0
[ 11.356658 0:8 axstarry::syscall:51] [syscall] id = 13, args = [3, 1073739808, 0, 8, 11056, 1], return 0
[ 11.360186 0:8 axstarry::syscall:38] [syscall] id = EXECVE, args = [6360, 6424, 6448, 11420, 3399988123389603631, 9259542123273814144], entry
[ 13.786608 0:8 axprocess::api:192] args: ["/lib64/ld-linux-x86-64.so.2", "./MediaServer", "-d"]
[ 13.789337 0:8 axprocess::api:194] The elf base addr may be different in different arch!
[ 13.791129 0:8 elf_parser:142] Base addr for the elf: 0x4000000
[ 13.792670 0:8 elf_parser:66] Base addr for the elf: 0x4000000
[ 13.794158 0:8 elf_parser::arch::x86_64:55] Base addr for the elf: 0x4000000
[ 13.795863 0:8 elf_parser::arch::x86_64:69] Relocating .rela.dyn
[ 13.797633 0:8 elf_parser::arch::x86_64:154] Relocating .rela.plt
[ 13.799085 0:8 elf_parser::arch::x86_64:178] Relocating done
[ 13.819216 0:8 axprocess::api:226] [new region] user heap: [VA:0x3fa00000, VA:0x3fe00000)
[ 13.820990 0:8 elf_parser::auxv:48] ELF header addr: 0x4000000
[ 13.833430 0:8 axprocess::api:245] [new region] user stack: [VA:0x3fe00000, VA:0x40000000)
[ 13.836721 0:8 axstarry::syscall:51] [syscall] id = 59, args = [6360, 6424, 6448, 11420, 3399988123389603631, 9259542123273814144], return 2
[ 13.843824 0:8 axstarry::syscall:20] [syscall] id = BRK, args = [0, 1073740732, 0, 35, 0, 0], entry
[ 13.847119 0:8 axstarry::syscall:51] [syscall] id = 12, args = [0, 1073740732, 0, 35, 0, 0], return 1067450368
[ 13.850437 0:8 axstarry::syscall:38] [syscall] id = ARCH_PRCTL, args = [12289, 1073740656, 67236800, 1, 3, 2048], entry
[ 13.853545 0:8 axstarry::syscall:51] [syscall] id = 158, args = [12289, 1073740656, 67236800, 1, 3, 2048], return -22
[ 13.856930 0:8 axstarry::syscall:29] [syscall] id = OPENAT, args = [4294967196, 67351264, 524288, 0, 524288, 67351264], entry
[ 13.860827 0:8 axstarry::syscall_fs::imp::io:407] path: "/MediaServer"
[ 13.873057 0:8 axprocess::link:243] create_link: /MediaServer → /MediaServer
[ 13.874599 0:8 axstarry::syscall:51] [syscall] id = 257, args = [4294967196, 67351264, 524288, 0, 524288, 67351264], return 4
[ 13.877181 0:8 axstarry::syscall:29] [syscall] id = READ, args = [4, 1073739128, 832, 0, 524288, 67351264], entry
[ 13.879606 0:8 axstarry::syscall_fs::imp::io:32] [read()] fd: 4, buf: 0x3ffff578, len: 832
[ 13.882102 0:8 axstarry::syscall:51] [syscall] id = 0, args = [4, 1073739128, 832, 0, 524288, 67351264], return 832
[ 13.884899 0:8 axstarry::syscall:29] [syscall] id = PREAD64, args = [4, 1073738112, 784, 64, 1073738112, 67351264], entry
[ 13.908408 0:8 axstarry::syscall:51] [syscall] id = 17, args = [4, 1073738112, 784, 64, 1073738112, 67351264], return 784
[ 13.911622 0:8 axstarry::syscall:29] [syscall] id = PREAD64, args = [4, 1073738048, 48, 880, 1073738112, 0], entry
[ 13.923982 0:8 axstarry::syscall:51] [syscall] id = 17, args = [4, 1073738048, 48, 880, 1073738112, 0], return 48
[ 13.926326 0:8 axstarry::syscall:29] [syscall] id = PREAD64, args = [4, 1073737968, 68, 928, 1073738112, 0], entry
[ 13.938559 0:8 axstarry::syscall:51] [syscall] id = 17, args = [4, 1073737968, 68, 928, 1073738112, 0], return 68
[ 13.941761 0:8 axstarry::syscall:38] [syscall] id = GETPID, args = [1073737284, 0, 67307088, 67307088, 67351264, 1], entry
[ 13.943868 0:8 axstarry::syscall:51] [syscall] id = 39, args = [1073737284, 0, 67307088, 67307088, 67351264, 1], return 7
[ 13.946927 0:8 axstarry::syscall:29] [syscall] id = WRITEV, args = [2, 1073737296, 6, 4294967295, 1073737209, 4294967295], entry
[ 13.949184 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3fffee44, len: 12
        7:     [ 13.951298 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x4030650, len: 5
file=[ 13.953399 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3ffffe21, len: 13
./MediaServer[ 13.955664 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x4030657, len: 2
 [[ 13.957594 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3fffedf8, len: 1
0[ 13.959539 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x403065c, len: 24
];  generating link map
```

不同1：

zlm

```
[ 16.687586 0:6 axstarry::syscall_task::imp::task:429] not support setpgid, try to set 7
[ 16.690262 0:6 axstarry::syscall:51] [syscall] id = 109, args = [7, 7, 0, 8, 11056, 1], return 0
[ 16.693719 0:6 axstarry::syscall:20] [syscall] id = MMAP, args = [0, 4096, 3, 34, 18446744073709551615, 0], entry
[ 16.697987 0:6 axmem:269] [mmap] vaddr: [VA:0x0, VA:0x1000), MappingFlags(READ | WRITE | USER), fixed: false, backend: false
[ 16.701751 0:6 axmem:287] find free area
[ 16.703343 0:6 axmem:292] found area [VA:0x6000, VA:0x7000)
[ 16.705622 0:6 axstarry::syscall:51] [syscall] id = 9, args = [0, 4096, 3, 34, 18446744073709551615, 0], return 24576
[ 16.708979 0:6 axhal::arch::x86_64::trap:21] User #PF @ 0x40c8f88, fault_vaddr=0x6008, error_code=0x4
[ 16.712026 0:6 axstarry::syscall:20] [syscall] id = MUNMAP, args = [24576, 4096, 4096, 1, 24576, 1], entry
[ 16.715053 0:6 axmem:310] [munmap] [VA:0x6000, VA:0x7000)
[ 16.716839 0:6 axmem:181] splitting for [VA:0x6000, VA:0x7000)
[ 16.718729 0:6 axmem:186]   drop [VA:0x6000, VA:0x7000)
[ 16.720599 0:6 axstarry::syscall:51] [syscall] id = 11, args = [24576, 4096, 4096, 1, 24576, 1], return 0
[ 16.723889 0:6 axstarry::syscall:29] [syscall] id = IOCTL, args = [0, 21505, 1073739960, 9196, 0, 2], entry
[ 16.727007 0:6 axstarry::syscall_fs::imp::ctl:438] fd: 0, request: 21505, argp: 1073739960
[ 16.729913 0:6 axprocess::stdio:123] stdin TCGETS | TIOCSPGRP, pretend to be tty.
[ 16.733201 0:6 axstarry::syscall:51] [syscall] id = 16, args = [0, 21505, 1073739960, 9196, 0, 2], return 0
[ 16.736702 0:6 axstarry::syscall:20] [syscall] id = MMAP, args = [0, 16384, 3, 34, 18446744073709551615, 0], entry
[ 16.740214 0:6 axmem:269] [mmap] vaddr: [VA:0x0, VA:0x4000), MappingFlags(READ | WRITE | USER), fixed: false, backend: false
[ 16.744637 0:6 axmem:287] find free area
[ 16.746644 0:6 axmem:292] found area [VA:0x6000, VA:0xa000)
[ 16.748909 0:6 axstarry::syscall:51] [syscall] id = 9, args = [0, 16384, 3, 34, 18446744073709551615, 0], return 24576
[ 16.752626 0:6 axhal::arch::x86_64::trap:21] User #PF @ 0x40c8f88, fault_vaddr=0x6008, error_code=0x4
[ 16.756360 0:6 axhal::arch::x86_64::trap:21] User #PF @ 0x40c864d, fault_vaddr=0x7080, error_code=0x6
```

sleep

```
[ 18.190262 0:6 axstarry::syscall_task::imp::task:429] not support setpgid, try to set 7
[ 18.192819 0:6 axstarry::syscall:51] [syscall] id = 109, args = [7, 7, 0, 8, 11056, 1], return 0
[ 18.196160 0:6 axstarry::syscall:29] [syscall] id = IOCTL, args = [0, 21505, 1073739960, 9196, 0, 2], entry
[ 18.199199 0:6 axstarry::syscall_fs::imp::ctl:438] fd: 0, request: 21505, argp: 1073739960
[ 18.201869 0:6 axprocess::stdio:123] stdin TCGETS | TIOCSPGRP, pretend to be tty.
[ 18.204472 0:6 axstarry::syscall:51] [syscall] id = 16, args = [0, 21505, 1073739960, 9196, 0, 2], return 0
[ 18.207581 0:6 axstarry::syscall:20] [syscall] id = MMAP, args = [0, 16384, 3, 34, 18446744073709551615, 0], entry
[ 18.210931 0:6 axmem:269] [mmap] vaddr: [VA:0x0, VA:0x4000), MappingFlags(READ | WRITE | USER), fixed: false, backend: false
[ 18.214731 0:6 axmem:287] find free area
[ 18.216335 0:6 axmem:292] found area [VA:0x6000, VA:0xa000)
[ 18.218169 0:6 axstarry::syscall:51] [syscall] id = 9, args = [0, 16384, 3, 34, 18446744073709551615, 0], return 24576
[ 18.221080 0:6 axhal::arch::x86_64::trap:21] User #PF @ 0x40c8f88, fault_vaddr=0x6008, error_code=0x4
[ 18.223721 0:6 axhal::arch::x86_64::trap:21] User #PF @ 0x40c864d, fault_vaddr=0x7070, error_code=0x6
```

不同2：

zlm

```
[ 19.672786 0:8 axstarry::syscall:51] [syscall] id = 20, args = [2, 1073737296, 6, 4294967295, 1073737209, 4294967295], return 57
[ 19.677989 0:8 axstarry::syscall:29] [syscall] id = GETCWD, args = [67352736, 128, 18446744073642198880, 4294967295, 536870912, 67352696], entry
[ 19.681952 0:8 axstarry::syscall:51] [syscall] id = 79, args = [67352736, 128, 18446744073642198880, 4294967295, 536870912, 67352696], return 67352736
[ 19.686604 0:8 axstarry::syscall:29] [syscall] id = PREAD64, args = [4, 1073737808, 784, 64, 536870912, 67352696], entry
[ 19.700459 0:8 axstarry::syscall:51] [syscall] id = 17, args = [4, 1073737808, 784, 64, 536870912, 67352696], return 784
[ 19.704326 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [0, 20314104, 1, 2050, 4, 0], entry
[ 19.707530 0:8 axmem:269] [mmap] vaddr: [VA:0x0, VA:0x1360000), MappingFlags(READ | USER), fixed: false, backend: true
[ 19.711728 0:8 axmem:287] find free area
[ 19.713503 0:8 axmem:292] found area [VA:0x1000, VA:0x1361000)
[ 19.716008 0:8 axstarry::syscall:51] [syscall] id = 9, args = [0, 20314104, 1, 2050, 4, 0], return 4096
[ 19.719024 0:8 axstarry::syscall:20] [syscall] id = MPROTECT, args = [10264576, 9883648, 0, 2050, 4, 0], entry
[ 19.723078 0:8 axmem:340] [mprotect] addr: [VA:0x9ca000, VA:0x1337000), flags: MappingFlags(USER)
[ 19.727046 0:8 axstarry::syscall:51] [syscall] id = 10, args = [10264576, 9883648, 0, 2050, 4, 0], return 0
[ 19.730931 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [10264576, 6320128, 5, 2066, 4, 10260480], entry
[ 19.734884 0:8 axmem:269] [mmap] vaddr: [VA:0x9ca000, VA:0xfd1000), MappingFlags(READ | EXECUTE | USER), fixed: true, backend: true
[ 19.738731 0:8 axmem:181] splitting for [VA:0x9ca000, VA:0xfd1000)
[ 19.741116 0:8 axmem:204]   shrink_left [VA:0x9ca000, VA:0x1337000) to [VA:0xfd1000, VA:0x1337000)
[ 19.744368 0:8 axstarry::syscall:51] [syscall] id = 9, args = [10264576, 6320128, 5, 2066, 4, 10260480], return 10264576
[ 19.748262 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [16584704, 3559424, 1, 2066, 4, 16580608], entry
[ 19.751684 0:8 axmem:269] [mmap] vaddr: [VA:0xfd1000, VA:0x1336000), MappingFlags(READ | USER), fixed: true, backend: true
[ 19.755631 0:8 axmem:181] splitting for [VA:0xfd1000, VA:0x1336000)
[ 19.758018 0:8 axmem:204]   shrink_left [VA:0xfd1000, VA:0x1337000) to [VA:0x1336000, VA:0x1337000)
[ 19.761006 0:8 axstarry::syscall:51] [syscall] id = 9, args = [16584704, 3559424, 1, 2066, 4, 16580608], return 16584704
[ 19.764625 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [20148224, 139264, 3, 2066, 4, 20140032], entry
[ 19.767833 0:8 axmem:269] [mmap] vaddr: [VA:0x1337000, VA:0x1359000), MappingFlags(READ | WRITE | USER), fixed: true, backend: true
[ 19.771543 0:8 axmem:181] splitting for [VA:0x1337000, VA:0x1359000)
[ 19.773604 0:8 axmem:204]   shrink_left [VA:0x1337000, VA:0x1361000) to [VA:0x1359000, VA:0x1361000)
[ 19.776842 0:8 axstarry::syscall:51] [syscall] id = 9, args = [20148224, 139264, 3, 2066, 4, 20140032], return 20148224
[ 19.780091 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x402b1da, fault_vaddr=0x1358760, error_code=0x6
[ 19.796847 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [20287488, 30712, 3, 50, 4294967295, 0], entry
[ 19.800202 0:8 axmem:269] [mmap] vaddr: [VA:0x1359000, VA:0x1361000), MappingFlags(READ | WRITE | USER), fixed: true, backend: false
[ 19.804968 0:8 axmem:181] splitting for [VA:0x1359000, VA:0x1361000)
[ 19.807336 0:8 axmem:186]   drop [VA:0x1359000, VA:0x1361000)
[ 19.810250 0:8 axstarry::syscall:51] [syscall] id = 9, args = [20287488, 30712, 3, 50, 4294967295, 0], return 20287488
[ 19.814219 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x4009562, fault_vaddr=0x1355f40, error_code=0x4
[ 19.830642 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x40095ad, fault_vaddr=0x1356000, error_code=0x4
[ 19.845999 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x400983b, fault_vaddr=0x1318, error_code=0x4
```

sleep

```
[ 18.569402 0:8 axstarry::syscall:51] [syscall] id = 20, args = [2, 1073737312, 6, 4294967295, 1073737225, 4294967295], return 51
[ 18.573629 0:8 axstarry::syscall:29] [syscall] id = GETCWD, args = [67352736, 128, 18446744073642198880, 4294967295, 536870912, 67352696], entry
[ 18.577891 0:8 axstarry::syscall:51] [syscall] id = 79, args = [67352736, 128, 18446744073642198880, 4294967295, 536870912, 67352696], return 67352736
[ 18.583039 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [0, 16408, 1, 2050, 4, 0], entry
[ 18.586022 0:8 axmem:269] [mmap] vaddr: [VA:0x0, VA:0x5000), MappingFlags(READ | USER), fixed: false, backend: true
[ 18.589720 0:8 axmem:287] find free area
[ 18.591251 0:8 axmem:292] found area [VA:0x1000, VA:0x6000)
[ 18.593403 0:8 axstarry::syscall:51] [syscall] id = 9, args = [0, 16408, 1, 2050, 4, 0], return 4096
[ 18.596713 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [8192, 4096, 5, 2066, 4, 4096], entry
[ 18.599782 0:8 axmem:269] [mmap] vaddr: [VA:0x2000, VA:0x3000), MappingFlags(READ | EXECUTE | USER), fixed: true, backend: true
[ 18.603905 0:8 axmem:181] splitting for [VA:0x2000, VA:0x3000)
[ 18.606337 0:8 axmem:191]   split [VA:0x1000, VA:0x6000) into 2 areas
[ 18.609192 0:8 axstarry::syscall:51] [syscall] id = 9, args = [8192, 4096, 5, 2066, 4, 4096], return 8192
[ 18.612510 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [12288, 4096, 1, 2066, 4, 8192], entry
[ 18.615600 0:8 axmem:269] [mmap] vaddr: [VA:0x3000, VA:0x4000), MappingFlags(READ | USER), fixed: true, backend: true
[ 18.619575 0:8 axmem:181] splitting for [VA:0x3000, VA:0x4000)
[ 18.620959 0:8 axmem:204]   shrink_left [VA:0x3000, VA:0x6000) to [VA:0x4000, VA:0x6000)
[ 18.623701 0:8 axstarry::syscall:51] [syscall] id = 9, args = [12288, 4096, 1, 2066, 4, 8192], return 12288
[ 18.627041 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [16384, 8192, 3, 2066, 4, 8192], entry
[ 18.630245 0:8 axmem:269] [mmap] vaddr: [VA:0x4000, VA:0x6000), MappingFlags(READ | WRITE | USER), fixed: true, backend: true
[ 18.633844 0:8 axmem:181] splitting for [VA:0x4000, VA:0x6000)
[ 18.635906 0:8 axmem:186]   drop [VA:0x4000, VA:0x6000)
[ 18.637992 0:8 axstarry::syscall:51] [syscall] id = 9, args = [16384, 8192, 3, 2066, 4, 8192], return 16384
[ 18.641433 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x402b260, fault_vaddr=0x5010, error_code=0x6
[ 18.645836 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x4009562, fault_vaddr=0x4dc0, error_code=0x4
[ 18.649392 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x400983b, fault_vaddr=0x12e0, error_code=0x4
```

不同3：

zlm

```
[ 19.943974 0:8 axstarry::syscall:51] [syscall] id = 20, args = [2, 1073736432, 18, 16, 1073736105, 4294967295], return 203
[ 19.948316 0:8 axstarry::syscall:38] [syscall] id = UNAME, args = [1073739600, 67111864, 0, 67271712, 0, 0], entry
[ 19.951814 0:8 axstarry::syscall:51] [syscall] id = 63, args = [1073739600, 67111864, 0, 67271712, 0, 0], return 0
[ 19.956858 0:8 axstarry::syscall:29] [syscall] id = ACCESS, args = [67304848, 4, 20275232, 0, 1, 67351280], entry
[ 19.960590 0:8 axstarry::syscall:51] [syscall] id = 21, args = [67304848, 4, 20275232, 0, 1, 67351280], return -2
[ 19.964699 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x402a837, fault_vaddr=0x1e8e6b, error_code=0x4
[ 19.980360 0:8 axstarry::syscall:38] [syscall] id = GETPID, args = [1073736020, 0, 67307488, 67307488, 0, 1], entry
[ 19.983742 0:8 axstarry::syscall:51] [syscall] id = 39, args = [1073736020, 0, 67307488, 67307488, 0, 1], return 7
[ 19.987257 0:8 axstarry::syscall:29] [syscall] id = WRITEV, args = [2, 1073736032, 12, 4294967295, 1073735897, 4294967295], entry
[ 19.991053 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3fffe954, len: 12
        7:     [ 19.994110 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x40307e0, len: 1
```

sleep

```
[ 18.733193 0:8 axstarry::syscall:51] [syscall] id = 20, args = [2, 1073737248, 18, 16, 1073736921, 4294967295], return 203
[ 18.738188 0:8 axstarry::syscall:38] [syscall] id = UNAME, args = [1073739616, 67111864, 0, 18446744073709547520, 0, 0], entry
[ 18.742388 0:8 axstarry::syscall:51] [syscall] id = 63, args = [1073739616, 67111864, 0, 18446744073709547520, 0, 0], return 0
[ 18.747942 0:8 axstarry::syscall:29] [syscall] id = ACCESS, args = [67304848, 4, 20032, 0, 0, 67351280], entry
[ 18.751797 0:8 axstarry::syscall:51] [syscall] id = 21, args = [67304848, 4, 20032, 0, 0, 67351280], return -2
[ 18.755494 0:8 axstarry::syscall:38] [syscall] id = GETPID, args = [1073736036, 0, 67307488, 67307488, 0, 1], entry
[ 18.758693 0:8 axstarry::syscall:51] [syscall] id = 39, args = [1073736036, 0, 67307488, 67307488, 0, 1], return 7
[ 18.762065 0:8 axstarry::syscall:29] [syscall] id = WRITEV, args = [2, 1073736048, 12, 4294967295, 1073735913, 4294967295], entry
[ 18.766100 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3fffe964, len: 12
        7:     [ 18.769332 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x40307e0, len: 1
```

不同4：

zlm

```
[ 20.157780 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3fffe7c4, len: 12
      7:     [ 20.160883 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x4030650, len: 5
file=[ 20.164070 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x1e8e6b, len: 11
libssl.so.3[ 20.167547 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x4030657, len: 2
 [[ 20.170446 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x3fffe778, len: 1
0[ 20.173156 0:8 axstarry::syscall_fs::imp::io:104] [write()] fd: 2, buf: 0x403065c, len: 24
];  generating link map
[ 20.176651 0:8 axstarry::syscall:51] [syscall] id = 20, args = [2, 1073735632, 6, 4294967295, 1073735545, 4294967295], return 55
[ 20.180416 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [0, 669680, 1, 2050, 4, 0], entry
[ 20.183420 0:8 axmem:269] [mmap] vaddr: [VA:0x0, VA:0xa4000), MappingFlags(READ | USER), fixed: false, backend: true
[ 20.186825 0:8 axmem:287] find free area
[ 20.188507 0:8 axmem:292] found area [VA:0x1361000, VA:0x1405000)
[ 20.190654 0:8 axstarry::syscall:51] [syscall] id = 9, args = [0, 669680, 1, 2050, 4, 0], return 20320256
[ 20.193890 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [20443136, 372736, 5, 2066, 4, 122880], entry
[ 20.197325 0:8 axmem:269] [mmap] vaddr: [VA:0x137f000, VA:0x13da000), MappingFlags(READ | EXECUTE | USER), fixed: true, backend: true
[ 20.201391 0:8 axmem:181] splitting for [VA:0x137f000, VA:0x13da000)
[ 20.203913 0:8 axmem:191]   split [VA:0x1361000, VA:0x1405000) into 2 areas
[ 20.206681 0:8 axstarry::syscall:51] [syscall] id = 9, args = [20443136, 372736, 5, 2066, 4, 122880], return 20443136
[ 20.210242 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [20815872, 118784, 1, 2066, 4, 495616], entry
[ 20.213426 0:8 axmem:269] [mmap] vaddr: [VA:0x13da000, VA:0x13f7000), MappingFlags(READ | USER), fixed: true, backend: true
[ 20.217037 0:8 axmem:181] splitting for [VA:0x13da000, VA:0x13f7000)
[ 20.219026 0:8 axmem:204]   shrink_left [VA:0x13da000, VA:0x1405000) to [VA:0x13f7000, VA:0x1405000)
[ 20.222157 0:8 axstarry::syscall:51] [syscall] id = 9, args = [20815872, 118784, 1, 2066, 4, 495616], return 20815872
[ 20.225927 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [20934656, 57344, 3, 2066, 4, 610304], entry
[ 20.229075 0:8 axmem:269] [mmap] vaddr: [VA:0x13f7000, VA:0x1405000), MappingFlags(READ | WRITE | USER), fixed: true, backend: true
[ 20.233172 0:8 axmem:181] splitting for [VA:0x13f7000, VA:0x1405000)
[ 20.235369 0:8 axmem:186]   drop [VA:0x13f7000, VA:0x1405000)
[ 20.237351 0:8 axstarry::syscall:51] [syscall] id = 9, args = [20934656, 57344, 3, 2066, 4, 610304], return 20934656
[ 20.240831 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x402b1da, fault_vaddr=0x14047b0, error_code=0x6
[ 20.244530 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x4009562, fault_vaddr=0x13ffe30, error_code=0x4
[ 20.247765 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x40095ad, fault_vaddr=0x1400000, error_code=0x4
[ 20.251123 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x400983b, fault_vaddr=0x1361270, error_code=0x4
```

sleep

```
[ 19.166765 0:8 axstarry::syscall:51] [syscall] id = 20, args = [2, 1073734784, 18, 16, 1073734457, 4294967295], return 203
[ 19.170357 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x402723a, fault_vaddr=0x243ce, error_code=0x4
[ 19.174369 0:8 axstarry::syscall:20] [syscall] id = MMAP, args = [0, 8192, 3, 34, 4294967295, 0], entry
[ 19.178377 0:8 axmem:269] [mmap] vaddr: [VA:0x0, VA:0x2000), MappingFlags(READ | WRITE | USER), fixed: false, backend: false
[ 19.182008 0:8 axmem:287] find free area
[ 19.183603 0:8 axmem:292] found area [VA:0x22f000, VA:0x231000)
[ 19.185600 0:8 axstarry::syscall:51] [syscall] id = 9, args = [0, 8192, 3, 34, 4294967295, 0], return 2289664
[ 19.189081 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x4003a66, fault_vaddr=0x22f000, error_code=0x6
[ 19.192999 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x4015f72, fault_vaddr=0x25d08, error_code=0x4
[ 19.196605 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x4015faa, fault_vaddr=0x26010, error_code=0x4
[ 19.200408 0:8 axhal::arch::x86_64::trap:21] User #PF @ 0x4014462, fault_vaddr=0x2300c0, error_code=0x6
```

MediaServer支持后台运行

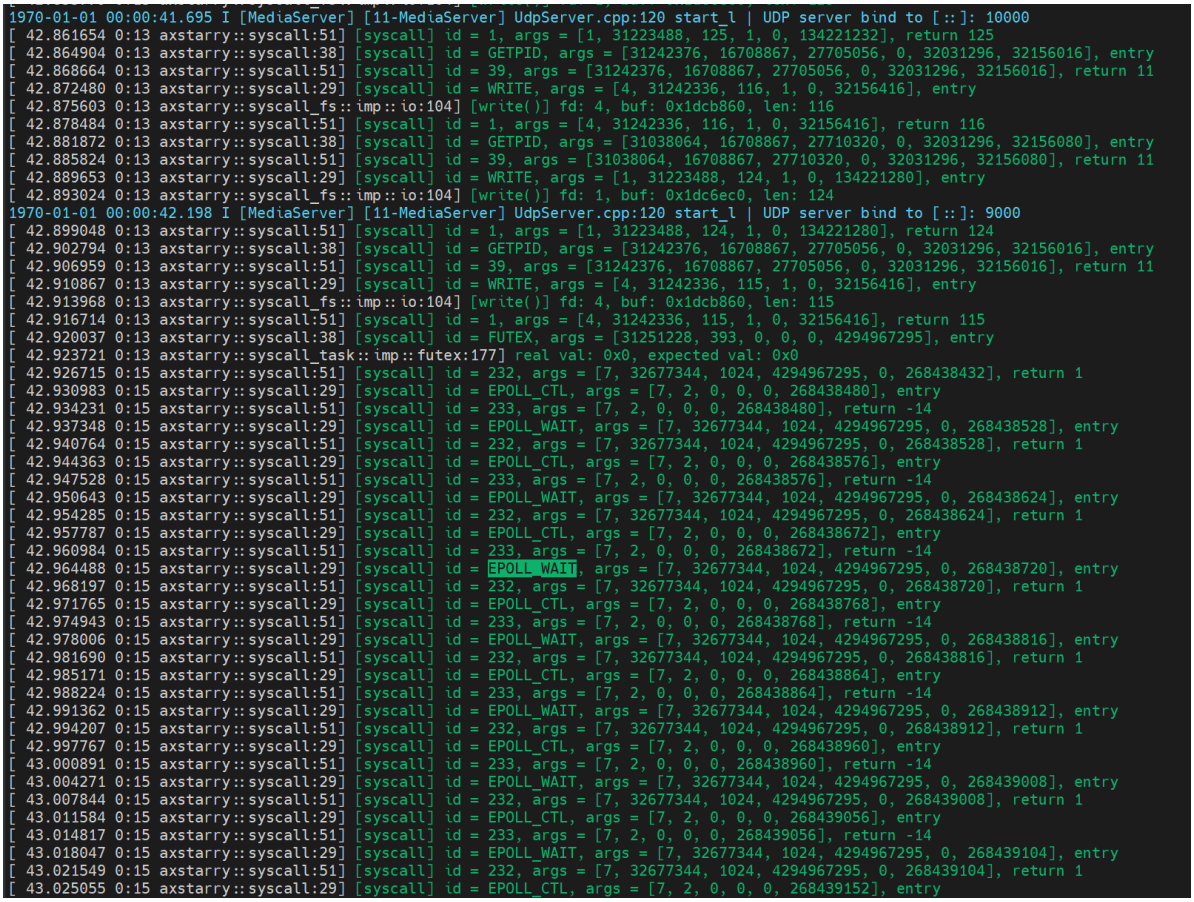
答：Starry原本支持&后台运行，但是由于ZLM的问题导致Starry出现死循环问题，由输入判断futex存在问题，futex_op为393，即

FUTEX_WAIT_BITSET | FUTEX_PRIVATE_FLAG | FUTEX_CLOCK_REALTIME

```
[syscall] id = 202, args = [31251224, 393, 0, 0, 0, 4294967295], return -22
[syscall] id = FUTEX, args = [31251224, 393, 0, 0, 0, 4294967295], entry
[syscall] id = 202, args = [31251224, 393, 0, 0, 0, 4294967295], return -22
[syscall] id = FUTEX, args = [31251224, 393, 0, 0, 0, 4294967295], entry
```

解决方法：取消判断`bitset`和`real_futex_val`的按位与。

修改后仍然不能后台运行zlm，从日志中看到zlm持续占用



原因：这个版本的Starry没有支持抢占，导致15号任务拿着cpu不能释放

解决方法：Starry分离了一个新版本，支持多核和抢占，将代码修改移植到这个版本中，开启抢占特性

拉取https://github.com/Starry-OS/Starry/代码后在main分支下先执行

```
cargo update
cargo update --precise 0.4.19 log
```

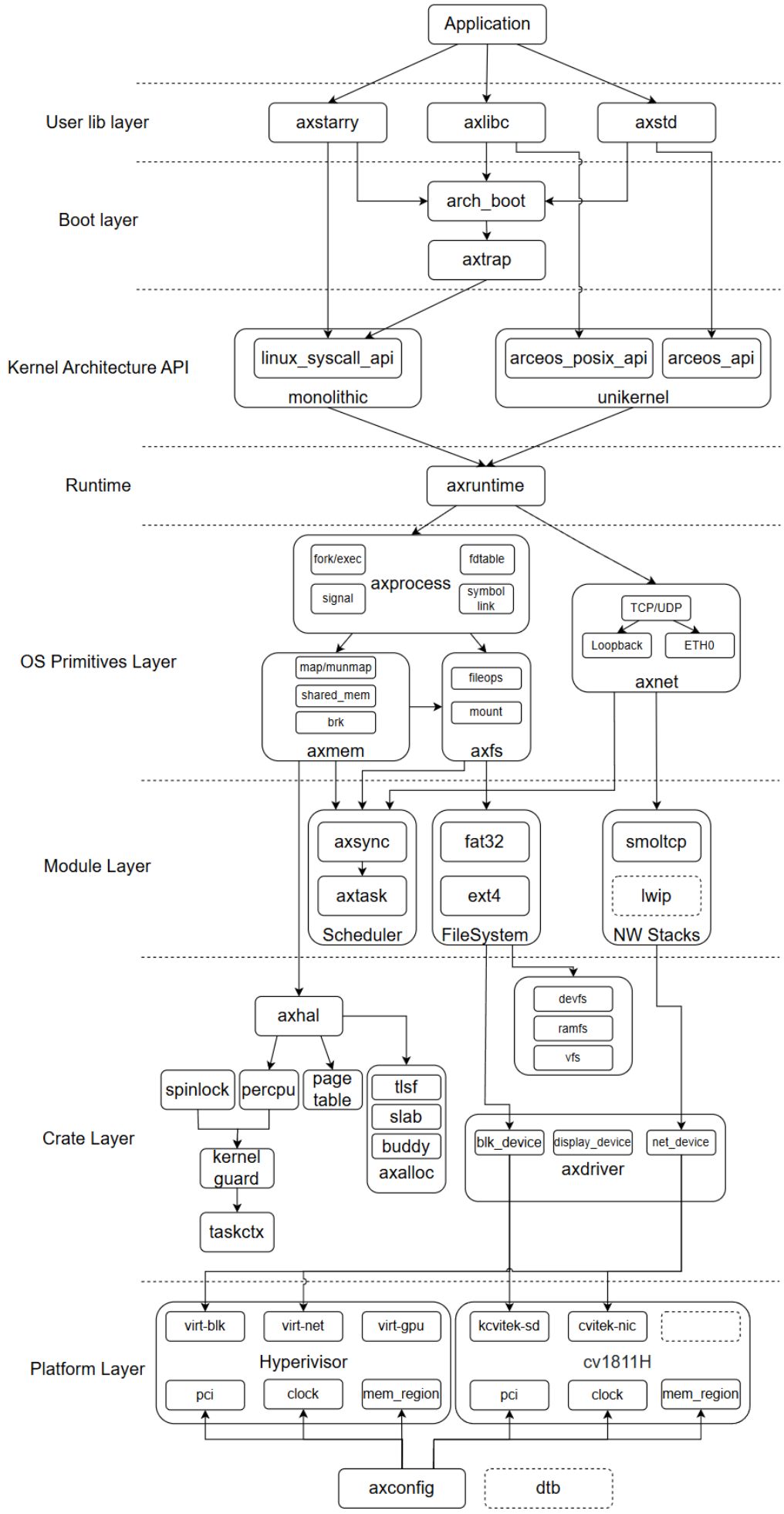
然后测试这个版本是否可运行

```
make A=apps/monolithic_userboot LOG=off ACCEL=n FEATURES=img,sched_rr
APP_FEATURES=batch ARCH=x86_64 run
```

musl，libc 的测试集合没有问题，但是到最后的 libc-test 的部分测例会出现卡住的问题，不过不影响运行。

根据需要添加指定的模块

Structure


Application
User lib layer
axstarry
axlibc
axstd
Boot layer
arch_boot
axtrap
Kernel Architecture API
linux_syscall_api
monolithic
arceos_posix_api
arceos_api
unikernel
Runtime
axruntime
OS Primitives Layer
fork/exec
fdtable
signal
axprocess
symbol link
TCP/UDP
Loopback
ETH0
axnet
map/munmap
shared_mem
brk
axmem
fileops
mount
axfs
Module Layer
axsync
axtask
Scheduler
fat32
ext4
FileSystem
smoltcp
lwip
NW Stacks
devfs
ramfs
vfs
axhal
Crate Layer
spinlock
percpu
page table
tlsf
slab
buddy
axalloc
kernel guard
taskctx
blk_device
display_device
net_device
axdriver
Platform Layer
virt-blk
virt-net
virt-gpu
Hyperivisor
pci
clock
mem_region
kcvitek-sd
cvitek-nic
cv1811H
pci
clock
mem_region
axconfig
dtb

ZLM的话添加

```
kbuild patch add arceos_posix_api  axruntime axfs axstarry linux_syscall_api
```

构建zlm的系统镜像

```
./build_img.sh -s 80 -file x86_64_ZLM -fs fat32
```

运行时加上multitask, sched_rr特性，其中img特性将virt-io切换成ramdisk，提高内核运行速度

```
make LOG=error FEATURES=img,multitask,sched_rr,sched_cfs run
```

后台执行MediaServer后效果如下图所示