实验 5: 驱动程序问题
管道驱动程序开发

魏旭，2015011015

2017 年 12 月 16 日

目录

1 运行环境

- Ubuntu 17.10
- Linux 4.13.0-19-generic
- GCC 7.2.0

2 程序结构

驱动程序模块整体分为三部分：

- 模块、设备初始化部分：创建两个字符设备，初始化缓存、互斥锁、信号量。
- 设备打开、关闭部分：描述`open()`、`close()`系统调用对设备的具体含义。等待读者、写者同时开始。
- 管道逻辑：描述`read()`、`write()`系统调用对设备的具体含义。实现阻塞式管道逻辑，当读写一方结束时结束另一方。

2.1 模块、设备初始化部分

mypipe.c

```c
static int mypipe_init(void) {
    int res;
    struct device *device;
    size_t i;
    char *dev_names[] = {MODULE_NAME"_in", MODULE_NAME"_out"};

    // 获得主设备号
    res = alloc_chrdev_region(&mypipe_devs[0], 0, 2, MODULE_NAME);
    if (res) {
        printk(KERN_ERR MODULE_NAME":alloc_chrdev_region error %d", res);
        goto fail;
    } else {
        printk(KERN_INFO MODULE_NAME":major number %d", MAJOR(mypipe_devs[0]));
    }
    mypipe_devs[1] = MKDEV(MAJOR(mypipe_devs[0]), 1);

    // 创建设备类
    cls = class_create(THIS_MODULE, MODULE_NAME);
    if (IS_ERR(cls)) {
        res = (int) PTR_ERR(cls);
        printk(KERN_ERR MODULE_NAME":class_create error %d", res);
        goto fail;
    }
    cls->devnode = mypipe_devnode;

    // 建立两个字符设备
    for (i = 0; i < 2; ++i) {
        cdev_init(&mypipe_cdevs[i], &fops[i]);
        res = cdev_add(&mypipe_cdevs[i], mypipe_devs[i], 1);
```

```
        if (res) {
            printk(KERN_ERR MODULE_NAME":cdev_add %zu error %d", i, res);
            goto fail;
        }
        device = device_create(cls, NULL, mypipe_devs[i], NULL, dev_names[i]);
        if (IS_ERR(device)) {
            res = (int) PTR_ERR(device);
            printk(KERN_ERR MODULE_NAME":device_create %zu error %d", i, res);
            goto fail;
        }
    }

    buffer = kmalloc(BUFFER_SIZE, GFP_KERNEL);
    if (!buffer) {
        printk(KERN_ERR MODULE_NAME":kmalloc error\n");
        res = -ENOMEM;
        goto fail;
    }

    mutex_init(&mutex_buffer);
    mutex_init(&mutex_occupied);

    sema_init(&sem_empty, 0);
    sema_init(&sem_full, 0);

    printk(KERN_INFO MODULE_NAME":inserted module\n");
    return 0;

    fail:
    mypipe_exit();
    return res;
}
```

设备初始化需要几步操作完成：

1. `alloc_chrdev_region()`分配设备号，结果保存在数组`mypipe_dev`中。注意*alloc_chrdev_region()*第一个参数返回首个设备的主次设备号，之后设备设备号需要计算（次设备号连续）。
2. `class_create()`建立设备类，之后设备都在这个类中。根据文档，返回的指针在错误时保存一个错误码，需要用宏*IS_ERR*检测、*PTR_ERR*转换。
3. `cdev_init()`创建字符设备结构`cdev`。确定设备号，以及设备文件支持的操作（在结构`file_operations`中）。
4. `cdev_add()`添加字符设备到系统。
5. `device_create()`将字符设备添加到设备类中，并给定一个名字。

之后初始化需要用到的互斥所和信号量。

注意到函数调用很多，所以检测了所有函数的返回值，方便 debug。一旦出错，调用`mypipe_exit()`回滚所有操作。*Linux* 内核文档建议用*goto*同意函数退出路径。所以用了标签*fail*

mypipe.c

```c
static void mypipe_exit(void) {
    size_t i;
    for (i = 0; i < 2; ++i) {
        device_destroy(cls, mypipe_devs[i]);
        cdev_del(&mypipe_cdevs[i]);
    }

    class_destroy(cls);
    unregister_chrdev_region(mypipe_devs[0], 2);

    kfree(buffer);

    mutex_destroy(&mutex_buffer);
    mutex_destroy(&mutex_occupied);

    printk(KERN_INFO MODULE_NAME ":removed module\n");
}
```

`mypipe_exit()`在模块卸载、模块安装出错时，回滚程序。

mypipe.c

```c
static char *mypipe_devnode(struct device *dev, umode_t *mode) {
    if (mode) {
        *mode = 0666;
    }
    return kasprintf(GFP_KERNEL, MODULE_NAME"/%s", dev_name(dev));
}
```

设备类中有 devtmpfs 文件系统回调函数，可以用来改变设备文件权限（默认为 600，改成 666，方便之后使用），并改变路径到 mypipe/下。

mypipe.c

```c
static struct file_operations fops[] = {
        {
                .owner=THIS_MODULE,
                .open=dev_open,
                .release=dev_release,
                .write= dev_write,
        },
        {
                .owner=THIS_MODULE,
                .open=dev_open,
                .release=dev_release,
                .read= dev_read,
        }
};
```

设备 0 不能读，设备 1 不能写，在文件可能操作上区分管道两端。

2.2 设备打开、关闭部分

mypipe.c

```c
static int dev_open(struct inode *inode, struct file *filp) {
    printk(KERN_INFO MODULE_NAME":attempting to open %u\n",
     ↪  MINOR(inode->i_cdev->dev));
```

```c
    mutex_lock_killable(&mutex_occupied);
    if (occupied[MINOR(inode->i_cdev->dev)]) {
        // 设备已被占用
        mutex_unlock(&mutex_occupied);
        return -EMFILE;
    } else {
        occupied[MINOR(inode->i_cdev->dev)] = true;
        if (occupied[0] && occupied[1]) {
            mutex_unlock(&mutex_occupied);
            // 读者、写着都准备好，可以开始
            up(&sem_empty);
        } else {
            mutex_unlock(&mutex_occupied);
            // 等待另一方
            down_killable(&sem_empty);
        }
        printk(KERN_INFO MODULE_NAME":opened %u\n", MINOR(inode->i_cdev->dev));
        return 0;
    }
}
```

open()打开设备文件时会调用这里定义的函数。查看 inode 中的次设备号inode->i_cdev->dev，保证读者、写者都准备好。这里借用了缓存“空”的信号*sem_empty*来同步读写，因为在读写都开始之前*sem_empty*是闲置的。

mypipe.c

```c
static int dev_release(struct inode *inode, struct file *filp) {
    printk(KERN_INFO MODULE_NAME":close %u\n", MINOR(inode->i_cdev->dev));
    mutex_lock_killable(&mutex_occupied);
    occupied[MINOR(inode->i_cdev->dev)] = false;
    if (!occupied[0] && !occupied[1]) {
        // 清空信号量、缓存
        sema_init(&sem_empty, 0);
        sema_init(&sem_full, 0);
        head = tail = 0;
    }
    mutex_unlock(&mutex_occupied);
    return 0;
}
```

close()关闭设备文件时会调用这里定义的函数。当两个设备都关闭时，清空缓冲区和信号量。

2.3 管道逻辑

mypipe.c

```c
static ssize_t dev_read(struct file *filp, char *buf, size_t count, loff_t *f_pos) {
    ssize_t res;
    size_t cnt;
    size_t i;
    printk(KERN_INFO MODULE_NAME":attempting to read %zu\n", count);

    mutex_lock_killable(&mutex_buffer);
```

```c
    while (head == tail) {
        mutex_lock_killable(&mutex_occupied);
        if (!occupied[0]) {
            mutex_unlock(&mutex_occupied);
            // 写者结束，终止读者
            printk(KERN_INFO MODULE_NAME":write closed %zu,%zu\n", head, tail);
            res = -EPIPE;
            goto fail;
        } else {
            mutex_unlock(&mutex_occupied);
            // 缓存空，等待写者
            printk(KERN_INFO MODULE_NAME":read empty %zu,%zu\n", head, tail);
            mutex_unlock(&mutex_buffer);
            up(&sem_full);
            down_killable(&sem_empty);
            mutex_lock_killable(&mutex_buffer);
        }
    }

    cnt = min(count, (BUFFER_SIZE + tail - head) % BUFFER_SIZE); // 计算实际读取字节，防止溢出
    printk(KERN_INFO MODULE_NAME":actually read %zu\n", cnt);
    for (i = 0; i < cnt; ++i) {
        res = put_user(buffer[(head + i) % BUFFER_SIZE], buf + i);
        if (res) {
            goto fail;
        }
    }

    if ((tail + 1) % BUFFER_SIZE == head) {
        // 缓存不满，写者可以继续
        up(&sem_full);
    }
    // 更新缓存头尾
    head = (head + cnt) % BUFFER_SIZE;
    printk(KERN_INFO MODULE_NAME":read %zu,%zu\n", head, tail);
    res = cnt;

    fail:
    mutex_unlock(&mutex_buffer);
    return res;
}
```

`read()`读设备文件时会调用这里定义的函数。这里处理几个逻辑：

- 缓存空时，如果写者已经结束，则读者必须结束；否则，等待写者缓存不空的信号。
- 缓存不空时，读取缓存。用户空间缓存大小`count`和缓存大小相比取小值，防止溢出。注意到参数*`buf`*是用户空间指针，需要用*`put_user()`*写入。
- 读取后，若发现缓存之前满了，则用信号量`sem_full`启动挂起的写者。

mypipe.c

```c
static ssize_t dev_write(struct file *filp, const char *buf, size_t count, loff_t *f_pos) {
    ssize_t res;
    size_t cnt;
    size_t i;
    printk(KERN_INFO MODULE_NAME":attempting to write %zu\n", count);

    mutex_lock_killable(&mutex_buffer);
    while ((tail + 1) % BUFFER_SIZE == head) {
        if (!occupied[1]) {
            mutex_unlock(&mutex_occupied);
            // 读者结束，终止写者
            printk(KERN_INFO MODULE_NAME":read closed %zu,%zu\n", head, tail);
            res = -EPIPE;
            goto fail;
        } else {
            mutex_unlock(&mutex_occupied);
            // 缓存满，等待读者
            printk(KERN_INFO MODULE_NAME":full %zu,%zu\n", head, tail);
            mutex_unlock(&mutex_buffer);
            up(&sem_empty);
            down_killable(&sem_full);
            mutex_lock_killable(&mutex_buffer);
        }
    }

    cnt = min(count, (BUFFER_SIZE + head - tail - 1) % BUFFER_SIZE); // 计算实际读取字节，防止溢出
    printk(KERN_INFO MODULE_NAME":actually write %zu\n", cnt);
    for (i = 0; i < cnt; ++i) {
        res = get_user(buffer[(tail + i) % BUFFER_SIZE], buf + i);
        if (res) {
            goto fail;
        }
    }

    if (head == tail) {
        // 缓存不空，读者可以继续
        up(&sem_empty);
    }
    // 更新缓存头尾
    tail = (tail + cnt) % BUFFER_SIZE;
    printk(KERN_INFO MODULE_NAME":write %zu,%zu\n", head, tail);
    res = cnt;

    fail:
    mutex_unlock(&mutex_buffer);
    return res;
}
```

`write()`写设备文件时会调用这里定义的函数。逻辑跟读类似。

3 运行情况

3.1 编译

Makefile

```
obj-m += mypipe.o
KERNELBUILD := /lib/modules/$(shell uname -r)/build
default:
        make -C $(KERNELBUILD) M=$(PWD) modules
clean:
        make -C $(KERNELBUILD) M=$(PWD) clean
```

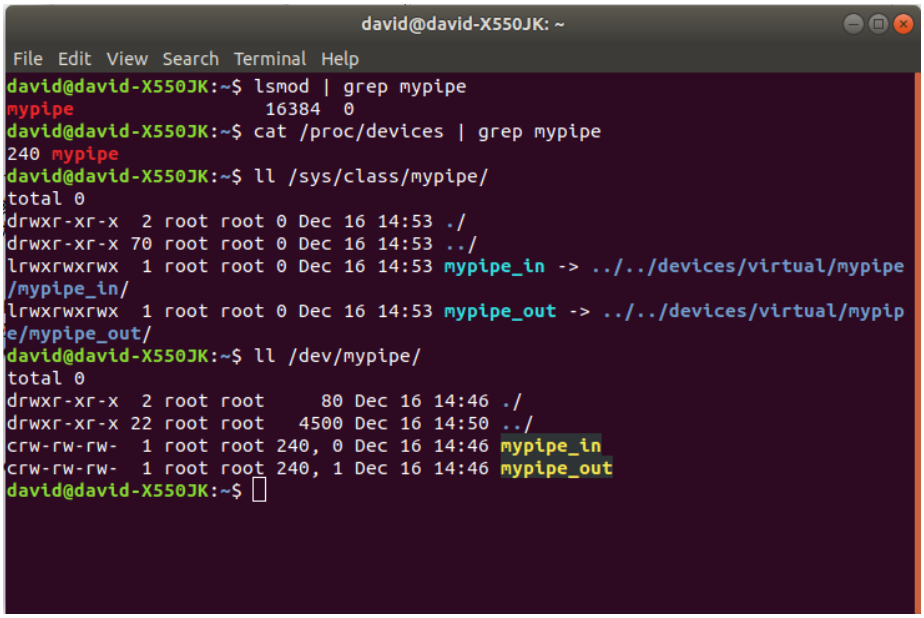
内核模块要用特殊的 makefile 编译，且内核版本要相同。最后输出内核模块文件 mypipe.ko, 用命令`sudo insmod mypipe.ko`安装模块。

3.2 安装

可以用一系列命令确认模块、设备安装成功：

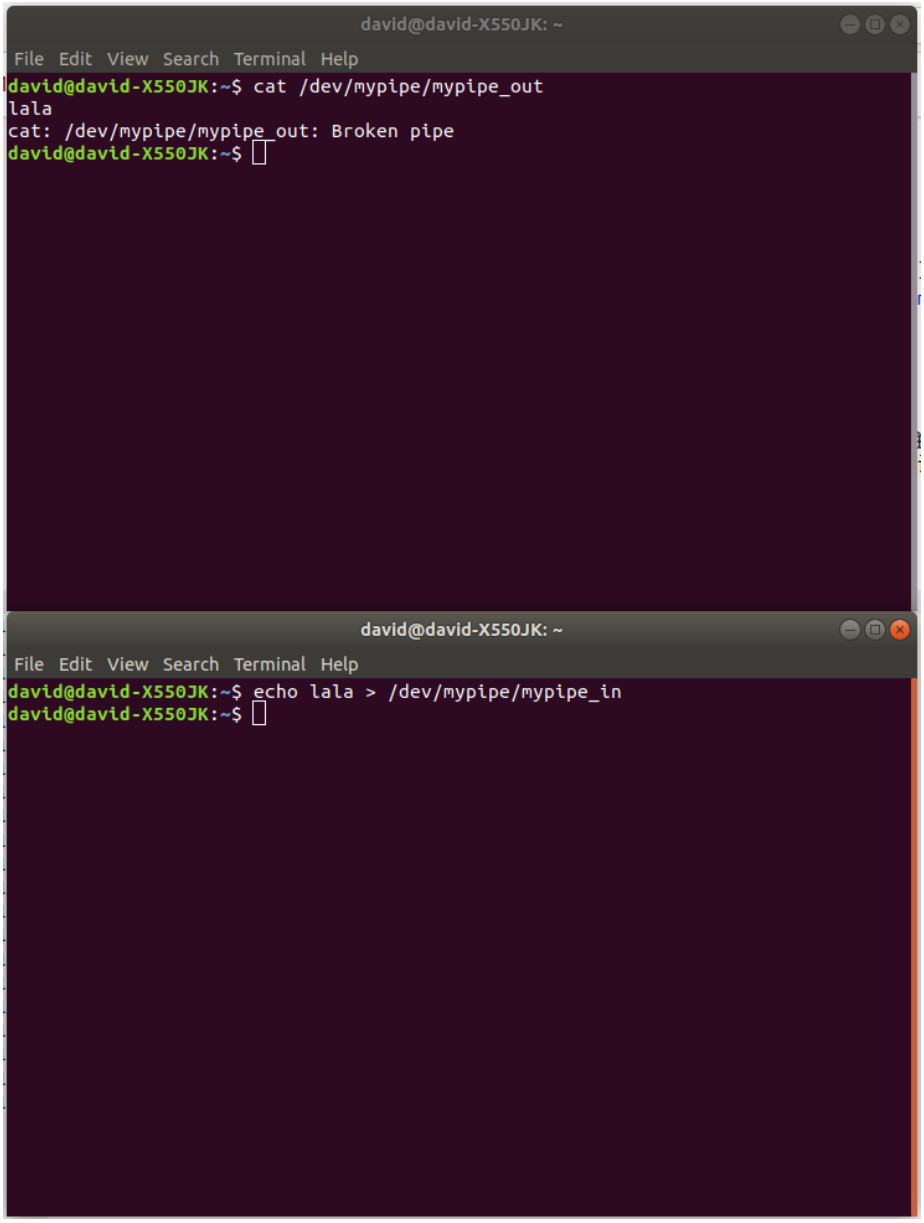
- 命令`lsmod | grep mypipe`可以看到模块安装成功。
- 命令`cat /proc/devices | grep mypipe`可以看到设备主设备号。
- 命令`ll /sys/class/mypipe/`可以看到`class_create()`注册的设备类。
- 命令`ll /dev/mypipe/`可以看到`device_create()`创建的设备文件。

结果如下

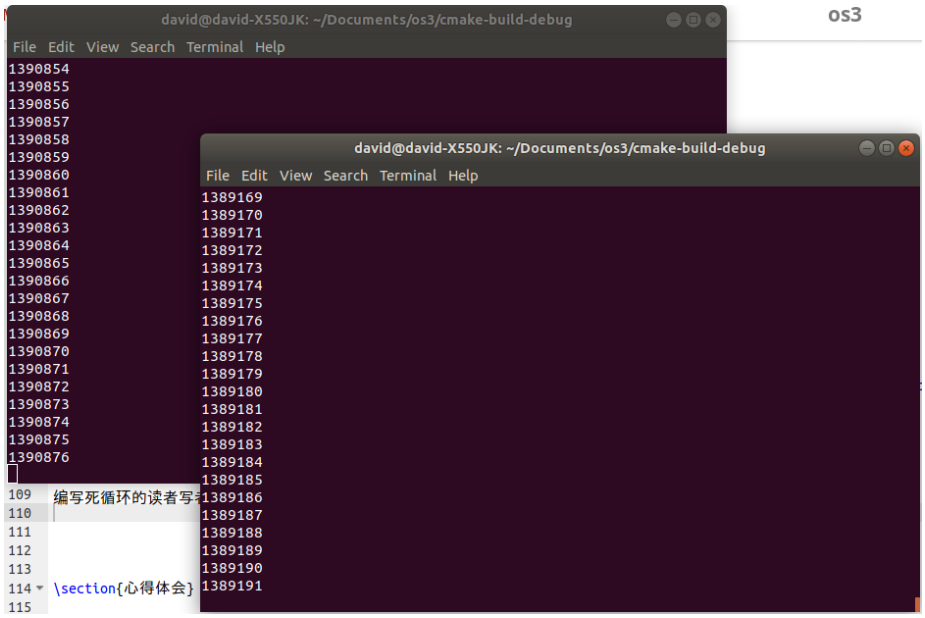


3.3 测试

在 bash 可以重定向的管道使命令工作。echo 写入结束后，cat 会在`read()`时会受到`-EPIPE`错误，如同在`dev_read()`中那样。



编写死循环的读者写者，正常工作。并且其中一个终止后，另一个也会终止。



3.4 系统日志

编写的内核模块中大量使用了`printk()`，所以系统日志里有大量输出，参加附录D

4 心得体会

- 系统级编程跟平时的应用级编程差距较大。我用的 CLion 平常用起来很方便。但遇到了内核编程风格时完全鬼畜，头文件都认为是错误的。
- Linux 内核模块编译需要自己写 Makefile。动态安装本质上是将模块二进制代码插入内核空间，共享了全局符号表。所以为了避免问题，所有自己的符号都是`static`的。
- 内核编程时用到了大量的宏和嵌套汇编，虽然看不太懂，但还是对 OS 内部运行机制有了第一手了解，很长见识。
- Linux 似乎是一个快速迭代但十分向后兼容的系统。就设备驱动这方面，在查资料时就发现了两种差距比较大的形式。一种是`register_chrdev()`和`mknod()`，直接注册设备号后手动建立设备文件；另一种是我采用的，步骤比较多，但是可以给予更多控制。结构`class`中成员`devnode`也是之后才有的，可以在驱动安装时配置好/dev 中的设备文件。
- 一切在 Linux 里都视为文件，设备类在/sys/class，设备在/dev，进程信息出现在/proc（似乎/sys 跟/proc 有重合）。通过自己写代码，对这种 Unix 哲学有了更深刻的感知。

- 系统编程需要区分内核空间和用户空间。大量用户控件显然的东西都不能使用，比如 stdio 和 pthread，需要利用近似的内核的库，有些不顺利。
- 不过不管是用户态还是内核态，并行编程大体思路是相通的。内核模块要提供系统调用的实现，而用户应用需求是并发的，所以内核模块需要同步不同部分。
- 这里实现了阻塞方式的管道，所以内核模块要处理读写同步的问题，所以使用了两个信号量表征缓存“空”和“满”，最终效果不错。

A 驱动模块代码

mypipe.c

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/device.h>
#include <linux/cdev.h>
#include <linux/slab.h>
#include <linux/uaccess.h>
#include <linux/errno.h>
#include <linux/mutex.h>
#include <linux/semaphore.h>

#define MODULE_NAME "mypipe"
#define BUFFER_SIZE ((size_t)100)


MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("Wei Xu");
MODULE_DESCRIPTION("Simple pipe implementation");

static struct class *cls;
static dev_t mypipe_devs[2];
static struct cdev mypipe_cdevs[2];
static bool occupied[2];
static struct mutex mutex_occupied;

static char *buffer;
static size_t head, tail;
static struct mutex mutex_buffer;
static struct semaphore sem_empty, sem_full;

static int dev_open(struct inode *inode, struct file *filp) {
    printk(KERN_INFO MODULE_NAME":attempting to open %u\n",
     ↪  MINOR(inode->i_cdev->dev));
    mutex_lock_killable(&mutex_occupied);
    if (occupied[MINOR(inode->i_cdev->dev)]) {
        // 设备已被占用
        mutex_unlock(&mutex_occupied);
        return -EMFILE;
    } else {
        occupied[MINOR(inode->i_cdev->dev)] = true;
        if (occupied[0] && occupied[1]) {
            mutex_unlock(&mutex_occupied);
            // 读者、写着都准备好，可以开始
            up(&sem_empty);
        } else {
            mutex_unlock(&mutex_occupied);
            // 等待另一方
            down_killable(&sem_empty);
        }
        printk(KERN_INFO MODULE_NAME":opened %u\n", MINOR(inode->i_cdev->dev));
        return 0;
    }
}

```

```c
static int dev_release(struct inode *inode, struct file *filp) {
    printk(KERN_INFO MODULE_NAME":close %u\n", MINOR(inode->i_cdev->dev));
    mutex_lock_killable(&mutex_occupied);
    occupied[MINOR(inode->i_cdev->dev)] = false;
    if (!occupied[0] && !occupied[1]) {
        // 清空信号量、缓存
        sema_init(&sem_empty, 0);
        sema_init(&sem_full, 0);
        head = tail = 0;
    }
    mutex_unlock(&mutex_occupied);
    return 0;
}

static ssize_t dev_read(struct file *filp, char *buf, size_t count, loff_t *f_pos) {
    ssize_t res;
    size_t cnt;
    size_t i;
    printk(KERN_INFO MODULE_NAME":attempting to read %zu\n", count);

    mutex_lock_killable(&mutex_buffer);
    while (head == tail) {
        mutex_lock_killable(&mutex_occupied);
        if (!occupied[0]) {
            mutex_unlock(&mutex_occupied);
            // 写者结束，终止读者
            printk(KERN_INFO MODULE_NAME":write closed %zu,%zu\n", head, tail);
            res = -EPIPE;
            goto fail;
        } else {
            mutex_unlock(&mutex_occupied);
            // 缓存空，等待写者
            printk(KERN_INFO MODULE_NAME":read empty %zu,%zu\n", head, tail);
            mutex_unlock(&mutex_buffer);
            up(&sem_full);
            down_killable(&sem_empty);
            mutex_lock_killable(&mutex_buffer);
        }
    }

    cnt = min(count, (BUFFER_SIZE + tail - head) % BUFFER_SIZE); // 计算实际读取字节，防止溢出
    printk(KERN_INFO MODULE_NAME":actually read %zu\n", cnt);
    for (i = 0; i < cnt; ++i) {
        res = put_user(buffer[(head + i) % BUFFER_SIZE], buf + i);
        if (res) {
            goto fail;
        }
    }

    if ((tail + 1) % BUFFER_SIZE == head) {
        // 缓存不满，写者可以继续
        up(&sem_full);
    }
    // 更新缓存头尾
    head = (head + cnt) % BUFFER_SIZE;
```

```
    printk(KERN_INFO MODULE_NAME":read %zu,%zu\n", head, tail);
    res = cnt;

    fail:
    mutex_unlock(&mutex_buffer);
    return res;
}

static ssize_t dev_write(struct file *filp, const char *buf, size_t count, loff_t *f_pos) {
    ssize_t res;
    size_t cnt;
    size_t i;
    printk(KERN_INFO MODULE_NAME":attempting to write %zu\n", count);

    mutex_lock_killable(&mutex_buffer);
    while ((tail + 1) % BUFFER_SIZE == head) {
        if (!occupied[1]) {
            mutex_unlock(&mutex_occupied);
            // 读者结束，终止写者
            printk(KERN_INFO MODULE_NAME":read closed %zu,%zu\n", head, tail);
            res = -EPIPE;
            goto fail;
        } else {
            mutex_unlock(&mutex_occupied);
            // 缓存满，等待读者
            printk(KERN_INFO MODULE_NAME":full %zu,%zu\n", head, tail);
            mutex_unlock(&mutex_buffer);
            up(&sem_empty);
            down_killable(&sem_full);
            mutex_lock_killable(&mutex_buffer);
        }
    }

    cnt = min(count, (BUFFER_SIZE + head - tail - 1) % BUFFER_SIZE); // 计算实际读取字节，防止溢出
    printk(KERN_INFO MODULE_NAME":actually write %zu\n", cnt);
    for (i = 0; i < cnt; ++i) {
        res = get_user(buffer[(tail + i) % BUFFER_SIZE], buf + i);
        if (res) {
            goto fail;
        }
    }

    if (head == tail) {
        // 缓存不空，读者可以继续
        up(&sem_empty);
    }
    // 更新缓存头尾
    tail = (tail + cnt) % BUFFER_SIZE;
    printk(KERN_INFO MODULE_NAME":write %zu,%zu\n", head, tail);
    res = cnt;

    fail:
    mutex_unlock(&mutex_buffer);
    return res;
}
```

```

static struct file_operations fops[] = {
        {
                .owner=THIS_MODULE,
                .open=dev_open,
                .release=dev_release,
                .write= dev_write,
        },
        {
                .owner=THIS_MODULE,
                .open=dev_open,
                .release=dev_release,
                .read= dev_read,
        }
};

static char *mypipe_devnode(struct device *dev, umode_t *mode) {
    if (mode) {
        *mode = 0666;
    }
    return kasprintf(GFP_KERNEL, MODULE_NAME"/%s", dev_name(dev));
}

static void mypipe_exit(void) {
    size_t i;
    for (i = 0; i < 2; ++i) {
        device_destroy(cls, mypipe_devs[i]);
        cdev_del(&mypipe_cdevs[i]);
    }

    class_destroy(cls);
    unregister_chrdev_region(mypipe_devs[0], 2);

    kfree(buffer);

    mutex_destroy(&mutex_buffer);
    mutex_destroy(&mutex_occupied);

    printk(KERN_INFO MODULE_NAME ":removed module\n");
}

static int mypipe_init(void) {
    int res;
    struct device *device;
    size_t i;
    char *dev_names[] = {MODULE_NAME"_in", MODULE_NAME"_out"};

    // 获得主设备号
    res = alloc_chrdev_region(&mypipe_devs[0], 0, 2, MODULE_NAME);
    if (res) {
        printk(KERN_ERR MODULE_NAME":alloc_chrdev_region error %d", res);
        goto fail;
    } else {
        printk(KERN_INFO MODULE_NAME":major number %d", MAJOR(mypipe_devs[0]));
    }
    mypipe_devs[1] = MKDEV(MAJOR(mypipe_devs[0]), 1);

```

```
    // 创建设备类
    cls = class_create(THIS_MODULE, MODULE_NAME);
    if (IS_ERR(cls)) {
        res = (int) PTR_ERR(cls);
        printk(KERN_ERR MODULE_NAME":class_create error %d", res);
        goto fail;
    }
    cls->devnode = mypipe_devnode;

    // 建立两个字符设备
    for (i = 0; i < 2; ++i) {
        cdev_init(&mypipe_cdevs[i], &fops[i]);
        res = cdev_add(&mypipe_cdevs[i], mypipe_devs[i], 1);
        if (res) {
            printk(KERN_ERR MODULE_NAME":cdev_add %zu error %d", i, res);
            goto fail;
        }
        device = device_create(cls, NULL, mypipe_devs[i], NULL, dev_names[i]);
        if (IS_ERR(device)) {
            res = (int) PTR_ERR(device);
            printk(KERN_ERR MODULE_NAME":device_create %zu error %d", i, res);
            goto fail;
        }
    }

    buffer = kmalloc(BUFFER_SIZE, GFP_KERNEL);
    if (!buffer) {
        printk(KERN_ERR MODULE_NAME":kmalloc error\n");
        res = -ENOMEM;
        goto fail;
    }

    mutex_init(&mutex_buffer);
    mutex_init(&mutex_occupied);

    sema_init(&sem_empty, 0);
    sema_init(&sem_full, 0);

    printk(KERN_INFO MODULE_NAME":inserted module\n");
    return 0;

    fail:
    mypipe_exit();
    return res;
}

module_init(mypipe_init);

module_exit(mypipe_exit);
```

B 写者代码

writer.c

```c
#include <stdio.h>

int main(void) {
    FILE *fp = fopen("/dev/mypipe/mypipe_in", "w");

    for (size_t i = 0; fprintf(fp, "%zu\n", i) >= 0; ++i) {
        printf("%zu\n", i);
    }
}
```

C 读者代码

reader.c

```c
#include <stdio.h>

int main(void) {
    FILE *fp = fopen("/dev/mypipe/mypipe_out", "r");

    size_t i;
    while (fscanf(fp, "%zu", &i) >= 0) {
        printf("%zu\n", i);
    }
}
```

D 系统日志输出（部分）

```
Dec 16 15:27:17 david-X550JK kernel: [11306.264239] mypipe:major number 240
Dec 16 15:27:17 david-X550JK kernel: [11306.264326] mypipe:inserted module
Dec 16 15:27:55 david-X550JK kernel: [11344.391854] mypipe:attempting to open 0
Dec 16 15:28:06 david-X550JK kernel: [11355.430383] mypipe:attempting to open 1
Dec 16 15:28:06 david-X550JK kernel: [11355.430388] mypipe:opened 1
Dec 16 15:28:06 david-X550JK kernel: [11355.430396] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.430397] mypipe:read empty 0,0
Dec 16 15:28:06 david-X550JK kernel: [11355.430399] mypipe:opened 0
Dec 16 15:28:06 david-X550JK kernel: [11355.431910] mypipe:attempting to write 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.431911] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.431914] mypipe:write 0,99
Dec 16 15:28:06 david-X550JK kernel: [11355.431915] mypipe:attempting to write 3997
Dec 16 15:28:06 david-X550JK kernel: [11355.431915] mypipe:full 0,99
Dec 16 15:28:06 david-X550JK kernel: [11355.431916] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.431918] mypipe:read 99,99
```

```
Dec 16 15:28:06 david-X550JK kernel: [11355.431919] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.431919] mypipe:write 99,98
Dec 16 15:28:06 david-X550JK kernel: [11355.431920] mypipe:attempting to write 3898
Dec 16 15:28:06 david-X550JK kernel: [11355.431920] mypipe:full 99,98
Dec 16 15:28:06 david-X550JK kernel: [11355.431921] mypipe:full 99,98
Dec 16 15:28:06 david-X550JK kernel: [11355.432052] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432053] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432055] mypipe:read 98,98
Dec 16 15:28:06 david-X550JK kernel: [11355.432057] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432059] mypipe:write 98,97
Dec 16 15:28:06 david-X550JK kernel: [11355.432062] mypipe:attempting to write 3799
Dec 16 15:28:06 david-X550JK kernel: [11355.432063] mypipe:full 98,97
Dec 16 15:28:06 david-X550JK kernel: [11355.432126] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432127] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432128] mypipe:read 97,97
Dec 16 15:28:06 david-X550JK kernel: [11355.432130] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432131] mypipe:write 97,96
Dec 16 15:28:06 david-X550JK kernel: [11355.432131] mypipe:attempting to write 3700
Dec 16 15:28:06 david-X550JK kernel: [11355.432132] mypipe:full 97,96
Dec 16 15:28:06 david-X550JK kernel: [11355.432199] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432199] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432201] mypipe:read 96,96
Dec 16 15:28:06 david-X550JK kernel: [11355.432202] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432203] mypipe:write 96,95
Dec 16 15:28:06 david-X550JK kernel: [11355.432204] mypipe:attempting to write 3601
Dec 16 15:28:06 david-X550JK kernel: [11355.432204] mypipe:full 96,95
Dec 16 15:28:06 david-X550JK kernel: [11355.432241] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432241] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432243] mypipe:read 95,95
Dec 16 15:28:06 david-X550JK kernel: [11355.432244] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432246] mypipe:write 95,94
Dec 16 15:28:06 david-X550JK kernel: [11355.432246] mypipe:attempting to write 3502
Dec 16 15:28:06 david-X550JK kernel: [11355.432247] mypipe:full 95,94
Dec 16 15:28:06 david-X550JK kernel: [11355.432282] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432283] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432284] mypipe:read 94,94
Dec 16 15:28:06 david-X550JK kernel: [11355.432285] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432286] mypipe:write 94,93
Dec 16 15:28:06 david-X550JK kernel: [11355.432287] mypipe:attempting to write 3403
Dec 16 15:28:06 david-X550JK kernel: [11355.432288] mypipe:full 94,93
Dec 16 15:28:06 david-X550JK kernel: [11355.432321] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432321] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432322] mypipe:read 93,93
Dec 16 15:28:06 david-X550JK kernel: [11355.432324] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432325] mypipe:write 93,92
Dec 16 15:28:06 david-X550JK kernel: [11355.432326] mypipe:attempting to write 3304
```

```
Dec 16 15:28:06 david-X550JK kernel: [11355.432326] mypipe:full 93,92
Dec 16 15:28:06 david-X550JK kernel: [11355.432361] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432361] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432363] mypipe:read 92,92
Dec 16 15:28:06 david-X550JK kernel: [11355.432364] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432365] mypipe:write 92,91
Dec 16 15:28:06 david-X550JK kernel: [11355.432366] mypipe:attempting to write 3205
Dec 16 15:28:06 david-X550JK kernel: [11355.432366] mypipe:full 92,91
Dec 16 15:28:06 david-X550JK kernel: [11355.432402] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432403] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432404] mypipe:read 91,91
Dec 16 15:28:06 david-X550JK kernel: [11355.432405] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432406] mypipe:write 91,90
Dec 16 15:28:06 david-X550JK kernel: [11355.432407] mypipe:attempting to write 3106
Dec 16 15:28:06 david-X550JK kernel: [11355.432408] mypipe:full 91,90
Dec 16 15:28:06 david-X550JK kernel: [11355.432445] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432446] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432447] mypipe:read 90,90
Dec 16 15:28:06 david-X550JK kernel: [11355.432449] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432450] mypipe:write 90,89
Dec 16 15:28:06 david-X550JK kernel: [11355.432450] mypipe:attempting to write 3007
Dec 16 15:28:06 david-X550JK kernel: [11355.432451] mypipe:full 90,89
Dec 16 15:28:06 david-X550JK kernel: [11355.432486] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432486] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432488] mypipe:read 89,89
Dec 16 15:28:06 david-X550JK kernel: [11355.432489] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432490] mypipe:write 89,88
Dec 16 15:28:06 david-X550JK kernel: [11355.432491] mypipe:attempting to write 2908
Dec 16 15:28:06 david-X550JK kernel: [11355.432492] mypipe:full 89,88
Dec 16 15:28:06 david-X550JK kernel: [11355.432526] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432526] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432528] mypipe:read 88,88
Dec 16 15:28:06 david-X550JK kernel: [11355.432529] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432530] mypipe:write 88,87
Dec 16 15:28:06 david-X550JK kernel: [11355.432531] mypipe:attempting to write 2809
Dec 16 15:28:06 david-X550JK kernel: [11355.432531] mypipe:full 88,87
Dec 16 15:28:06 david-X550JK kernel: [11355.432572] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432573] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432575] mypipe:read 87,87
Dec 16 15:28:06 david-X550JK kernel: [11355.432576] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432577] mypipe:write 87,86
Dec 16 15:28:06 david-X550JK kernel: [11355.432579] mypipe:attempting to write 2710
Dec 16 15:28:06 david-X550JK kernel: [11355.432579] mypipe:full 87,86
Dec 16 15:28:06 david-X550JK kernel: [11355.432632] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432633] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432634] mypipe:read 86,86
```

```
Dec 16 15:28:06 david-X550JK kernel: [11355.432636] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432638] mypipe:write 86,85
Dec 16 15:28:06 david-X550JK kernel: [11355.432640] mypipe:attempting to write 2611
Dec 16 15:28:06 david-X550JK kernel: [11355.432640] mypipe:full 86,85
Dec 16 15:28:06 david-X550JK kernel: [11355.432682] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432683] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432684] mypipe:read 85,85
Dec 16 15:28:06 david-X550JK kernel: [11355.432685] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432686] mypipe:write 85,84
Dec 16 15:28:06 david-X550JK kernel: [11355.432687] mypipe:attempting to write 2512
Dec 16 15:28:06 david-X550JK kernel: [11355.432688] mypipe:full 85,84
Dec 16 15:28:06 david-X550JK kernel: [11355.432722] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432722] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432723] mypipe:read 84,84
Dec 16 15:28:06 david-X550JK kernel: [11355.432725] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432726] mypipe:write 84,83
Dec 16 15:28:06 david-X550JK kernel: [11355.432727] mypipe:attempting to write 2413
Dec 16 15:28:06 david-X550JK kernel: [11355.432727] mypipe:full 84,83
Dec 16 15:28:06 david-X550JK kernel: [11355.432764] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432764] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432765] mypipe:read 83,83
Dec 16 15:28:06 david-X550JK kernel: [11355.432766] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432767] mypipe:write 83,82
Dec 16 15:28:06 david-X550JK kernel: [11355.432768] mypipe:attempting to write 2314
Dec 16 15:28:06 david-X550JK kernel: [11355.432768] mypipe:full 83,82
Dec 16 15:28:06 david-X550JK kernel: [11355.432804] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432805] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432806] mypipe:read 82,82
Dec 16 15:28:06 david-X550JK kernel: [11355.432807] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432808] mypipe:write 82,81
Dec 16 15:28:06 david-X550JK kernel: [11355.432808] mypipe:attempting to write 2215
Dec 16 15:28:06 david-X550JK kernel: [11355.432809] mypipe:full 82,81
Dec 16 15:28:06 david-X550JK kernel: [11355.432847] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432847] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432848] mypipe:read 81,81
Dec 16 15:28:06 david-X550JK kernel: [11355.432849] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432850] mypipe:write 81,80
Dec 16 15:28:06 david-X550JK kernel: [11355.432851] mypipe:attempting to write 2116
Dec 16 15:28:06 david-X550JK kernel: [11355.432852] mypipe:full 81,80
Dec 16 15:28:06 david-X550JK kernel: [11355.432888] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432888] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432889] mypipe:read 80,80
Dec 16 15:28:06 david-X550JK kernel: [11355.432890] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432891] mypipe:write 80,79
Dec 16 15:28:06 david-X550JK kernel: [11355.432892] mypipe:attempting to write 2017
Dec 16 15:28:06 david-X550JK kernel: [11355.432893] mypipe:full 80,79
```

```
Dec 16 15:28:06 david-X550JK kernel: [11355.432931] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432932] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432933] mypipe:read 79,79
Dec 16 15:28:06 david-X550JK kernel: [11355.432934] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432935] mypipe:write 79,78
Dec 16 15:28:06 david-X550JK kernel: [11355.432936] mypipe:attempting to write 1918
Dec 16 15:28:06 david-X550JK kernel: [11355.432936] mypipe:full 79,78
Dec 16 15:28:06 david-X550JK kernel: [11355.432972] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.432972] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432973] mypipe:read 78,78
Dec 16 15:28:06 david-X550JK kernel: [11355.432974] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.432975] mypipe:write 78,77
Dec 16 15:28:06 david-X550JK kernel: [11355.432976] mypipe:attempting to write 1819
Dec 16 15:28:06 david-X550JK kernel: [11355.432976] mypipe:full 78,77
Dec 16 15:28:06 david-X550JK kernel: [11355.433013] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433014] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433014] mypipe:read 77,77
Dec 16 15:28:06 david-X550JK kernel: [11355.433015] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433016] mypipe:write 77,76
Dec 16 15:28:06 david-X550JK kernel: [11355.433017] mypipe:attempting to write 1720
Dec 16 15:28:06 david-X550JK kernel: [11355.433018] mypipe:full 77,76
Dec 16 15:28:06 david-X550JK kernel: [11355.433053] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433054] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433055] mypipe:read 76,76
Dec 16 15:28:06 david-X550JK kernel: [11355.433056] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433057] mypipe:write 76,75
Dec 16 15:28:06 david-X550JK kernel: [11355.433057] mypipe:attempting to write 1621
Dec 16 15:28:06 david-X550JK kernel: [11355.433058] mypipe:full 76,75
Dec 16 15:28:06 david-X550JK kernel: [11355.433096] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433097] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433098] mypipe:read 75,75
Dec 16 15:28:06 david-X550JK kernel: [11355.433099] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433100] mypipe:write 75,74
Dec 16 15:28:06 david-X550JK kernel: [11355.433101] mypipe:attempting to write 1522
Dec 16 15:28:06 david-X550JK kernel: [11355.433101] mypipe:full 75,74
Dec 16 15:28:06 david-X550JK kernel: [11355.433138] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433138] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433139] mypipe:read 74,74
Dec 16 15:28:06 david-X550JK kernel: [11355.433140] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433141] mypipe:write 74,73
Dec 16 15:28:06 david-X550JK kernel: [11355.433142] mypipe:attempting to write 1423
Dec 16 15:28:06 david-X550JK kernel: [11355.433143] mypipe:full 74,73
Dec 16 15:28:06 david-X550JK kernel: [11355.433181] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433181] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433182] mypipe:read 73,73
Dec 16 15:28:06 david-X550JK kernel: [11355.433183] mypipe:actually write 99
```

```
Dec 16 15:28:06 david-X550JK kernel: [11355.433185] mypipe:write 73,72
Dec 16 15:28:06 david-X550JK kernel: [11355.433186] mypipe:attempting to write 1324
Dec 16 15:28:06 david-X550JK kernel: [11355.433186] mypipe:full 73,72
Dec 16 15:28:06 david-X550JK kernel: [11355.433220] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433220] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433222] mypipe:read 72,72
Dec 16 15:28:06 david-X550JK kernel: [11355.433223] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433224] mypipe:write 72,71
Dec 16 15:28:06 david-X550JK kernel: [11355.433224] mypipe:attempting to write 1225
Dec 16 15:28:06 david-X550JK kernel: [11355.433225] mypipe:full 72,71
Dec 16 15:28:06 david-X550JK kernel: [11355.433261] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433262] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433263] mypipe:read 71,71
Dec 16 15:28:06 david-X550JK kernel: [11355.433264] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433265] mypipe:write 71,70
Dec 16 15:28:06 david-X550JK kernel: [11355.433266] mypipe:attempting to write 1126
Dec 16 15:28:06 david-X550JK kernel: [11355.433266] mypipe:full 71,70
Dec 16 15:28:06 david-X550JK kernel: [11355.433303] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433304] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433305] mypipe:read 70,70
Dec 16 15:28:06 david-X550JK kernel: [11355.433306] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433307] mypipe:write 70,69
Dec 16 15:28:06 david-X550JK kernel: [11355.433308] mypipe:attempting to write 1027
Dec 16 15:28:06 david-X550JK kernel: [11355.433309] mypipe:full 70,69
Dec 16 15:28:06 david-X550JK kernel: [11355.433346] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433346] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433347] mypipe:read 69,69
Dec 16 15:28:06 david-X550JK kernel: [11355.433348] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433349] mypipe:write 69,68
Dec 16 15:28:06 david-X550JK kernel: [11355.433350] mypipe:attempting to write 928
Dec 16 15:28:06 david-X550JK kernel: [11355.433351] mypipe:full 69,68
Dec 16 15:28:06 david-X550JK kernel: [11355.433388] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433389] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433390] mypipe:read 68,68
Dec 16 15:28:06 david-X550JK kernel: [11355.433391] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433392] mypipe:write 68,67
Dec 16 15:28:06 david-X550JK kernel: [11355.433392] mypipe:attempting to write 829
Dec 16 15:28:06 david-X550JK kernel: [11355.433393] mypipe:full 68,67
Dec 16 15:28:06 david-X550JK kernel: [11355.433430] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433431] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433432] mypipe:read 67,67
Dec 16 15:28:06 david-X550JK kernel: [11355.433433] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433434] mypipe:write 67,66
Dec 16 15:28:06 david-X550JK kernel: [11355.433434] mypipe:attempting to write 730
Dec 16 15:28:06 david-X550JK kernel: [11355.433435] mypipe:full 67,66
Dec 16 15:28:06 david-X550JK kernel: [11355.433468] mypipe:attempting to read 4096
```

```
Dec 16 15:28:06 david-X550JK kernel: [11355.433468] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433469] mypipe:read 66,66
Dec 16 15:28:06 david-X550JK kernel: [11355.433470] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433471] mypipe:write 66,65
Dec 16 15:28:06 david-X550JK kernel: [11355.433472] mypipe:attempting to write 631
Dec 16 15:28:06 david-X550JK kernel: [11355.433472] mypipe:full 66,65
Dec 16 15:28:06 david-X550JK kernel: [11355.433503] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433503] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433504] mypipe:read 65,65
Dec 16 15:28:06 david-X550JK kernel: [11355.433505] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433506] mypipe:write 65,64
Dec 16 15:28:06 david-X550JK kernel: [11355.433507] mypipe:attempting to write 532
Dec 16 15:28:06 david-X550JK kernel: [11355.433507] mypipe:full 65,64
Dec 16 15:28:06 david-X550JK kernel: [11355.433533] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433533] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433534] mypipe:read 64,64
Dec 16 15:28:06 david-X550JK kernel: [11355.433535] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433536] mypipe:write 64,63
Dec 16 15:28:06 david-X550JK kernel: [11355.433537] mypipe:attempting to write 433
Dec 16 15:28:06 david-X550JK kernel: [11355.433537] mypipe:full 64,63
Dec 16 15:28:06 david-X550JK kernel: [11355.433566] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433566] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433567] mypipe:read 63,63
Dec 16 15:28:06 david-X550JK kernel: [11355.433568] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433569] mypipe:write 63,62
Dec 16 15:28:06 david-X550JK kernel: [11355.433570] mypipe:attempting to write 334
Dec 16 15:28:06 david-X550JK kernel: [11355.433571] mypipe:full 63,62
Dec 16 15:28:06 david-X550JK kernel: [11355.433597] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433597] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433598] mypipe:read 62,62
Dec 16 15:28:06 david-X550JK kernel: [11355.433599] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433600] mypipe:write 62,61
Dec 16 15:28:06 david-X550JK kernel: [11355.433601] mypipe:attempting to write 235
Dec 16 15:28:06 david-X550JK kernel: [11355.433601] mypipe:full 62,61
Dec 16 15:28:06 david-X550JK kernel: [11355.433630] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433630] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433631] mypipe:read 61,61
Dec 16 15:28:06 david-X550JK kernel: [11355.433632] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433633] mypipe:write 61,60
Dec 16 15:28:06 david-X550JK kernel: [11355.433633] mypipe:attempting to write 136
Dec 16 15:28:06 david-X550JK kernel: [11355.433634] mypipe:full 61,60
Dec 16 15:28:06 david-X550JK kernel: [11355.433658] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433658] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433659] mypipe:read 60,60
Dec 16 15:28:06 david-X550JK kernel: [11355.433660] mypipe:actually write 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433661] mypipe:write 60,59
```

```
Dec 16 15:28:06 david-X550JK kernel: [11355.433662] mypipe:attempting to write 37
Dec 16 15:28:06 david-X550JK kernel: [11355.433662] mypipe:full 60,59
Dec 16 15:28:06 david-X550JK kernel: [11355.433688] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433689] mypipe:actually read 99
Dec 16 15:28:06 david-X550JK kernel: [11355.433690] mypipe:read 59,59
Dec 16 15:28:06 david-X550JK kernel: [11355.433691] mypipe:actually write 37
Dec 16 15:28:06 david-X550JK kernel: [11355.433691] mypipe:write 59,96
Dec 16 15:28:06 david-X550JK kernel: [11355.433731] mypipe:attempting to read 4096
Dec 16 15:28:06 david-X550JK kernel: [11355.433732] mypipe:actually read 37
Dec 16 15:28:06 david-X550JK kernel: [11355.433733] mypipe:read 96,96
```