```
#include <linux/kfifo.h>
```

- create a fifo

```
/**
 * kfifo_alloc - dynamically allocates a new fifo buffer
 * @fifo: pointer to the fifo
 * @size: the number of elements in the fifo, this must be a power of 2
 * @gfp_mask: get_free_pages mask, passed to kmalloc()
 *
 * This macro dynamically allocates a new fifo buffer.
 *
 * The numer of elements will be rounded-up to a power of 2.
 * The fifo will be release with kfifo_free().
 * Return 0 if no error, otherwise an error code.
 */
#define kfifo_alloc(fifo, size, gfp_mask) \
```

```
    static struct kfifo cmd_fifo;

    ret = kfifo_alloc(&cmd_fifo, PAGE_SIZE, GFP_KERNEL);
    if (ret)
    {
        printk(KERN_ERR "kfifo alloc failed\n");
    }
```

- put data into fifo

```
static void dec_fuser_post_cmd(int cmd)
{
    kfifo_in(&cmd_fifo, &cmd, sizeof(cmd));
    up(&queue_sem);
}
```

- retrieve data from fifo

```
        int value;

        ret = kfifo_out(&cmd_fifo, &value, sizeof(value));
        if (ret != sizeof(value))
        {
            printk("no pending value\n");
        }
        if (value == 0xffffffff)
        {
            printk("Got the termination command\n");
            kfifo_free(&cmd_fifo);
           // kthread_stop(task);
            break;
        }
```

- free fifo

  > kfifo_free(&cmd_fifo);