```
/**
 * simple_read_from_buffer - copy data from the buffer to user space
 * @to: the user space buffer to read to
 * @count: the maximum number of bytes to read
 * @ppos: the current position in the buffer
 * @from: the buffer to read from
 * @available: the size of the buffer
 *
 * The simple_read_from_buffer() function reads up to @count bytes from the
 * buffer @from at offset @ppos into the user space address starting at @to.
 *
 * On success, the number of bytes read is returned and the offset @ppos is
 * advanced by this number, or negative value is returned on error.
 **/
ssize_t simple_read_from_buffer(void __user *to, size_t count,
                        loff_t *ppos, const void *from, size_t available);
```

sample:

```
static ssize_t
reset_pin_dbgfs_read(struct file *file, char __user *user_buf,
                                                size_t count, loff_t *ppos)
{
        int reset_pin = (int)file->private_data;
        int size = 0;
        char *buff;
        ssize_t ret;
        int level;
        BUG_ON(!gpio_is_valid(reset_pin));

        buff = kmalloc(100, GFP_KERNEL);
        BUG_ON(!buff);

        level = gpio_get_value(reset_pin);

        size = sprintf(buff, "current reset-pin is %u\n", level);

        ret = simple_read_from_buffer(user_buf, count, ppos, buff, size);
        kfree(buff);

        return ret;
}
```

把kernel data 放置到user space，由read interface带回。

/**

 * simple_write_to_buffer - copy data from user space to the buffer

 * @to: the buffer to write to

 * @available: the size of the buffer

 * @ppos: the current position in the buffer

 * @from: the user space buffer to read from

 * @count: the maximum number of bytes to read

 *

 * The simple_write_to_buffer() function reads up to @count bytes from the user

 * space address starting at @from into the buffer @to at offset @ppos.

```
*

* On success, the number of bytes written is returned and the offset @ppos is

* advanced by this number, or negative value is returned on error.

**/

ssize_t simple_write_to_buffer(void *to, size_t available, loff_t *ppos,

                const void __user *from, size_t count);
```