ioctl

Advanced Embedded Software Development

with **Dan Walkes**

University of Colorado **Boulder**

Learning objectives:

Understand a kernel ioctl and its purpose.

Understand how to write an ioctl handler for a kernel driver.

ioctl

- ioctl is a system call on a file descriptor which is passed to the driver.
- Identifies a command to be performed and another argument (typically a pointer)
- Requires another program (or python script?) to issue through system call.

ioctl

- Common interface used for device control
- Request for a device other than read/write, for example
  - Lock a door
  - eject media
  - report information
- User space prototype:

```
#include <sys/ioctl.h>

int ioctl(int fd, unsigned long request, ...);
```

Linux Device Drivers 3rd Edition Chapter 6
http://man7.org/linux/man-pages/man2/ioctl.2.html

ioctl

```
#include <sys/ioctl.h>

int ioctl(int fd, unsigned long request, ...);
```

- What do the dots mean?
  - Typically represent varargs
  - In this case, are a single optional argument argp
  - request argument describes argp use

> The second argument is a device-dependent request code. The third argument is an untyped pointer to memory. It's traditionally **char** ***argp*** (from the days before **void** * was valid C), and will be so named for this discussion.

Linux Device Drivers 3rd Edition Chapter 6
http://man7.org/linux/man-pages/man2/ioctl.2.html

ioctl

- Easiest and most straightforward choice for device operations
- Has fallen “out of favor among kernel developers”
  - Unstructured system calls
    - difficult to audit
    - often not well documented
- Alternatives
  - embedding commands into the data stream
  - Using virtual filesystems like sysfs

ioctl

- file operations table signature
  - Has changed in 2005 from the type mentioned in the book
  - inode argument referenced in the book available at filp->f_inode

```
int (*ioctl) (struct inode *inode, [illegible] file *filp,
              unsigned int cmd, unsi[illegible]long arg);
```

```
long (*unlocked_ioctl) (struct file *filp, unsigned int cmd,
                        unsigned long arg);
```

```
struct file {
    struct inode        *f_inode;
}
```

```
struct file_operations {
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char __user *,
    ssize_t (*write) (struct file *, const char __u
    long (*unlocked_ioctl) (struct file *, unsigned int, un
    long (*compat_ioctl) (struct file *, unsigned int, unsi
}
```

Linux Device Drivers 3rd Edition Chapter 6
https://elixir.bootlin.com/linux/v5.3.1/source/include/linux/fs.h#L1789
https://lwn.net/Articles/119652/
https://elixir.bootlin.com/linux/v5.3.1/source/include/linux/fs.h#L928

ioctl

- “The optional arg argument is passed in the form of an unsigned long, regardless of whether it was given by the user as an integer or pointer”
- What is the problem with this approach?
  - Assumes sizeof(unsigned long) = sizeof(void *)
- New implementation includes compat_ioctl to handle differences in arg size between 64 and 32 bit
  - Avoids breaking 32 bit programs running on 64 bit kernels with assumptions about arg size

```
long (*compat_ioctl) (struct file *filp, unsigned int cmd,
                      unsigned long arg);
```

Linux Device Drivers 3rd Edition Chapter 6
https://lwn.net/Articles/119652/

ioctl example

- Switch statement based on cmd argument

```
long scull_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
{
switch(cmd) {

  case SCULL_IOCRESET:
        scull_quantum = SCULL_QUANTUM;
        scull_qset = SCULL_QSET;
        break;

  case SCULL_IOCSQUANTUM: /* Set: arg points to the value */
        if (! capable (CAP_SYS_ADMIN))
                return -EPERM;
        retval = __get_user(scull_quantum, (int __user *)arg);
        break;
```

```
#define SCULL_IOCRESET    _IO(SCULL_IOC_MAGIC, 0)
```

```
#define SCULL_IOCSQUANTUM _IOW(SCULL_IOC_MAGIC,  1, int)
```

Linux Device Drivers 3rd Edition Chapter 6
https://github.com/cu-ecen-5013/ldd3/blob/scullc-5-ubuntu/scull/main.c#L394
https://github.com/cu-ecen-5013/ldd3/blob/scullc-5-ubuntu/scull/scull.h#L152

Selecting ioctl cmd numbers

- Macros help us generate unique cmd codes for each device driver using “magic” numbers specific to driver
  - Avoids issuing the correct command to the wrong device

```
/* Use 'k' as magic number */
#define SCULL_IOC_MAGIC  'k'
```

```
#define SCULL_IOCRESET    _IO(SCULL_IOC_MAGIC, 0)
```

Code	Seq# (hex)	Include File	Comments
'k'	00-0F	linux/spi/spidev.h	conflict!
'k'	00-05	video/kyro.h	conflict!
'k'	10-17	linux/hsi/hsi_char.h	HSI character device

The first argument to _IO, _IOW, _IOR, or _IOWR is an identifying letter or number from the table below. Because of the large number of drivers, many drivers share a partial letter with other drivers.

Linux Device Drivers 3rd Edition Chapter 6
https://github.com/torvalds/linux/blob/master/Documentation/ioctl/ioctl-number.rst
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/scull/scull.h

Using _IO macros

- First argument - magic number
- Second arg - sequence (command) number
- 3rd arg - type of data transferred

```
#define _IOW(type,nr,size)        _IOC(_IOC_WRITE,(type),(nr),(_IOC_TYPECHECK(size)))
```

```
#define _IOC(dir,type,nr,size) \
        (((dir)  << _IOC_DIRSHIFT) | \
         ((type) << _IOC_TYPESHIFT) | \
         ((nr)   << _IOC_NRSHIFT) | \
         ((size) << _IOC_SIZESHIFT))
```

```
#define SCULL_IOCRESET     _IO(SCULL_IOC_MAGIC, 0)
```

```
#define SCULL_IOCSQUANTUM _IOW(SCULL_IOC_MAGIC,  1, int)
```

_IO	an	ioctl with no parameters
_IOW	an	ioctl with write parameters (copy_from_user)
_IOR	an	ioctl with read parameters (copy_to_user)
_IOWR	an	ioctl with both write and read parameters.

Linux Device Drivers 3rd Edition Chapter 6
https://elixir.bootlin.com/linux/v5.3.1/source/include/uapi/asm-generic/ioctl.h#L85
https://github.com/torvalds/linux/blob/master/Documentation/ioctl/ioctl-number.rst

Using _IO macros

- _IOC_READ (_IOR) - transfer from kernel to user space
- _IOC_WRITE (_IOW)- transfer from user to kernel space

```
#define _IOW(type,nr,size)        _IOC(_IOC_WRITE,(type),(nr),(_IOC_TYPECHECK(size)))
```

```
#define _IOC(dir,type,nr,size) \
        (((dir)  << _IOC_DIRSHIFT) | \
         ((type) << _IOC_TYPESHIFT) | \
         ((nr)   << _IOC_NRSHIFT) | \
         ((size) << _IOC_SIZESHIFT))
```

```
#define SCULL_IOCRESET     _IO(SCULL_IOC_MAGIC, 0)
```

```
#define SCULL_IOCSQUANTUM _IOW(SCULL_IOC_MAGIC,  1, int)
```

_IO	an	ioctl with no parameters
_IOW	an	ioctl with write parameters (copy_from_user)
_IOR	an	ioctl with read parameters (copy_to_user)
_IOWR	an	ioctl with both write and read parameters.

Linux Device Drivers 3rd Edition Chapter 6
https://elixir.bootlin.com/linux/v5.3.1/source/include/uapi/asm-generic/ioctl.h#L85
https://github.com/torvalds/linux/blob/master/Documentation/ioctl/ioctl-number.rst

Predefined commands

- Some ioctls are recognized by the kernel, decoded before passing to your driver
- Avoid magic type “T” to avoid these

Code	Seq# (hex)	Include File	Comments
'T'	all	arch/x86/include/asm/ioctls.h	conflict!

Linux Device Drivers 3rd Edition Chapter 6
https://github.com/torvalds/linux/blob/master/Documentation/ioctl/ioctl-number.rst

write() as ioctl alternative

- Write control sequences to the device
  - Doesn’t require a program (ioctl) to implement
    - echo “start” > /dev/yourdevice
  - Especially useful for devices which don’t transfer data
  - May make the driver more complex in some cases
    - if so ioctl or sysfs may be a better solution

Linux Device Drivers 3rd Edition Chapter 6