Gabriele **Proietti Mattia**

Advanced Operating Systems and Virtualization

[Lab 08] Misc devices, `ioctl` and `/proc` filesystem

DIAG
Department of Computer, Control and Management Engineering "A. Ruberti", Sapienza University of Rome

gpm.name · proiettimattia@diag.uniroma1.it

A.Y. 2020/2021 · v1

Introduction

Examples are from the folder

https://github.com/gabrielepmattia/aosv-code-examples/tree/main/08-devices

Char and Misc devices

DIAG

Creating a char device

For creating a char device you can use the function

```
static inline int register_chrdev(unsigned int major, const char *name,
                                  const struct file_operations *fops)
```

By passing the structure that contains the file_operations. Remember that this will not create automatically a special file in /dev but you will need to create it manually. This can be done:

- within the kernel module (suggested)
- from userspace

If you pass 0, your new device will be assigned to the first free major number

Miscellaneous Devices

There is a special kind of char devices which are called misc devices. They are meant to be used just for generic purposes. APIs are

```
79  struct miscdevice  {
80          int minor;
81          const char *name;
82          const struct file_operations *fops;
83          struct list_head list;
84          struct device *parent;
85          struct device *this_device;
86          const struct attribute_group **groups;
87          const char *nodename;
88          umode_t mode;
89  };
90
91  extern int misc_register(struct miscdevice *misc);
92  extern void misc_deregister(struct miscdevice *misc);
```

https://elixir.bootlin.com/linux/v5.11/source/include/linux/miscdevice.h#L91

ioctl

ioctl

The ioctl is a special system call, as read or write which can be issued to a file to trigger special actions.

```
#include <sys/ioctl.h>
int ioctl(int fd, unsigned long request, ...);
```

DESCRIPTION

The ioctl() system call manipulates the underlying device parameters of special files. In particular, many operating characteristics of character special files (e.g., terminals) may be controlled with ioctl() requests. **The argument fd must be an open file descriptor.**

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.

An ioctl() request has encoded in it whether the argument is an in parameter or out parameter, and the size of the argument argp in bytes. Macros and defines used in specifying an ioctl() request are located in the file <sys/ioctl.h>.

File operations

```
1820  struct file_operations {
1821          struct module *owner;
1822          loff_t (*llseek) (struct file *, loff_t, int);
1823          ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
1824          ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
1825          ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
1826          ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
1827          int (*iopoll)(struct kiocb *kiocb, bool spin);
1828          int (*iterate) (struct file *, struct dir_context *);
1829          int (*iterate_shared) (struct file *, struct dir_context *);
1830          __poll_t (*poll) (struct file *, struct poll_table_struct *);
1831          long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
1832          long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
1833          int (*mmap) (struct file *, struct vm_area_struct *);
1834          unsigned long mmap_supported_flags;
1835          int (*open) (struct inode *, struct file *);
1836          int (*flush) (struct file *, fl_owner_t id);
```

https://elixir.bootlin.com/linux/v5.11/source/include/linux/fs.h#L1820

Further info https://lwn.net/Articles/119652/

/proc

Creating a /proc entry

You can create a file in /proc by using the function:

```
struct proc_dir_entry *proc_create(const char *name, umode_t mode,
                                   struct proc_dir_entry *parent,
                                   const struct proc_ops *proc_ops);
```

Remember to pass `proc_ops` instead of `file_operations`.

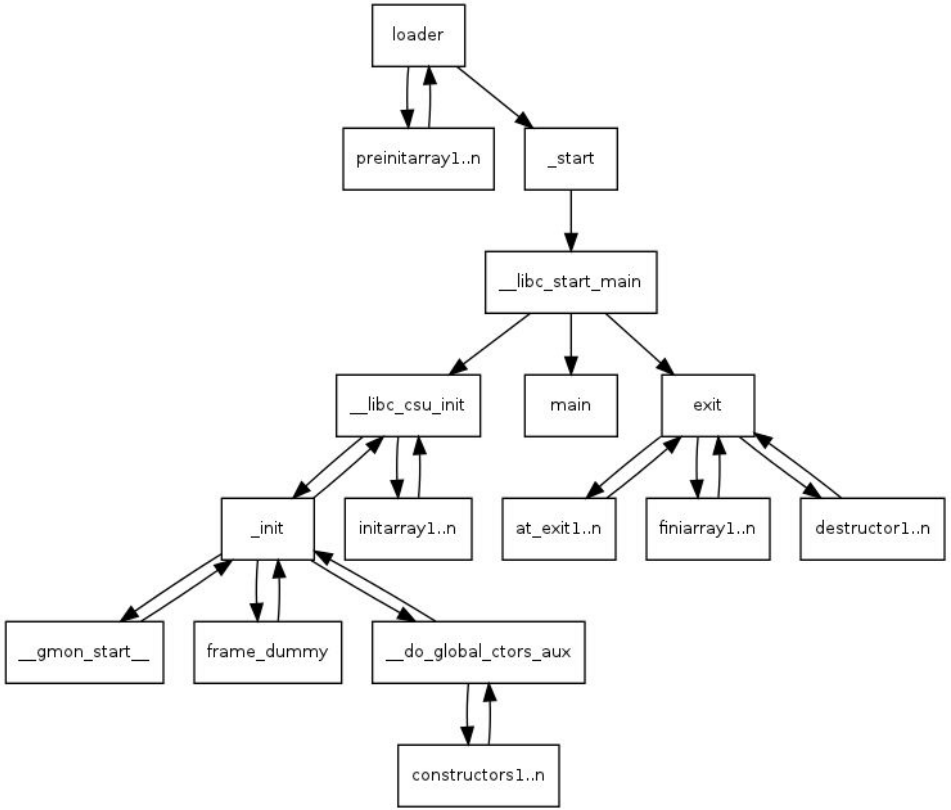
```
29  struct proc_ops {
30          unsigned int proc_flags;
31          int     (*proc_open)(struct inode *, struct file *);
32          ssize_t (*proc_read)(struct file *, char __user *, size_t, loff_t *);
33          ssize_t (*proc_read_iter)(struct kiocb *, struct iov_iter *);
34          ssize_t (*proc_write)(struct file *, const char __user *, size_t, loff_t *);
35          loff_t  (*proc_lseek)(struct file *, loff_t, int);
36          int     (*proc_release)(struct inode *, struct file *);
37          __poll_t (*proc_poll)(struct file *, struct poll_table_struct *);
38          long    (*proc_ioctl)(struct file *, unsigned int, unsigned long);
39  #ifdef CONFIG_COMPAT
40          long    (*proc_compat_ioctl)(struct file *, unsigned int, unsigned long);
41  #endif
42          int     (*proc_mmap)(struct file *, struct vm_area_struct *);
43          unsigned long (*proc_get_unmapped_area)(struct file *, unsigned long, unsigned
44  } __randomize_layout;
```

https://elixir.bootlin.com/linux/v5.11/source/include/linux/proc_fs.h#L29

Constructors in C

GCC Wrapper

The libc adds a set of facilities to the user space code. When you write and compile a C program with GCC the call graph is the following.



Advanced Operating Systems and Virtualization

[Lab 08] Misc devices, `ioctl` and `/proc` filesystem

LECTURER

Gabriele **Proietti Mattia**

gpm.name · proiettimattia@diag.uniroma1.it

DIAG