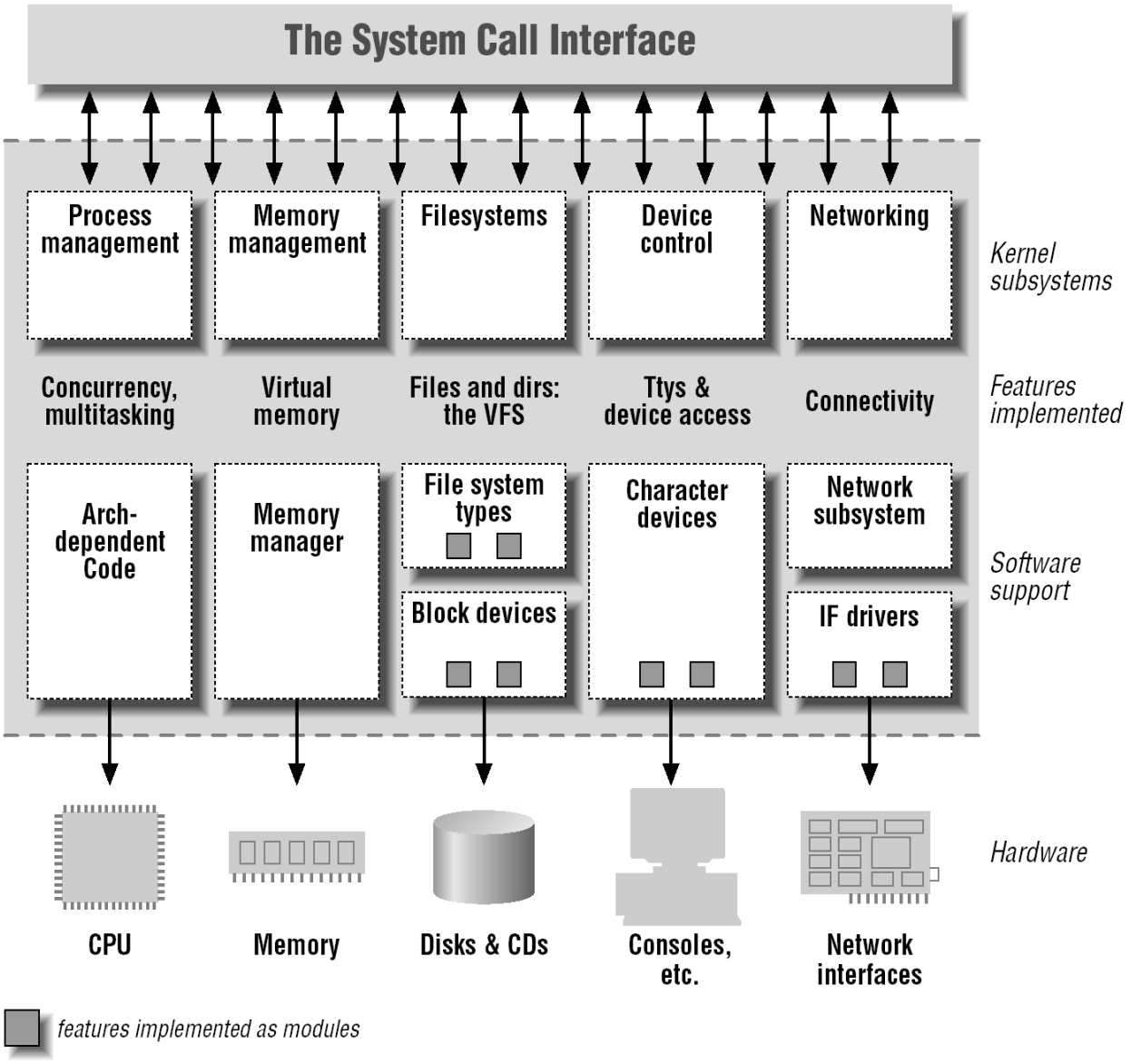
Unix (Linux) Device Drivers

- Kernel module that handles the interaction with an specific hardware device, hiding its operational details behind a common interface
- Three basic categories
  - Character
  - Block
  - Network

Kernel Overview: LINUX



Hardware Devices

- Accessible via `/dev` pseudo-files
- Kernel redirect pseudo-file operations to proper device driver services considering *major* and *minor* numbers
  - *Major*
    - Identifies a driver within the kernel (8 bits)
  - *Minor*
    - Identifies a device (unit) within the driver

Char Devices

- Byte streams
  (e.g. `/dev/console`, `/dev/ttyS0`, `/dev/st0`)
- Operate mostly like ordinary files
  - No backward seeks

Block Devices

- Block-accessible devices at I/O level
- File system related devices (e.g. disks)
- Share a common interface with char devices, but distinct semantics
  - Block oriented (accessing single bytes is a waste)
  - Seekable
- Additional operations to support file systems

Net Devices

- Do not fit properly under the pseudo-file interface
  - Usually not a node in a file system
  - Integration with a protocol stack
- Generic network interface instead
  - Communication related operations (e.g. sending, receiving, package marshaling, time-out handling, statistic collection)
  - Optimized for TCP/IP integration

Hello World Module

```
[root]#cat > hello.c
#define MODULE
#include <linux/module.h>
int init_module(){printk("Hello World!"); return 0;}
void cleanup_module(){printk("Good Bye!");}
^D
[root]# gcc -c hello.c
[root]# insmod hello.o
[root]# dmesg
[root]# rmmod hello
[root]# dmesg
```

Module Initialization

- Initialization

```
int init_module(void)
```

  - Module's entry point
  - Called at loading (by insmod)
  - Performs module registration

- Finalization

```
void cleanup_module(void)
```

  - Module's exit point
  - Called at unloading (by rmmod)
  - Performs module unregistration

Module Registration

- Binds a module to the kernel's `syscall` interface
- Registration

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

- Unregistration

```
int unregister_chrdev(unsigned int major,
const char *name)
```

- Pseudo file

```
mknod /dev/devname0 c major minor
```

Module Parameters

- Externally accessible module-global variables
- Declared via MODULE_PARM macro

```
int irq = 10;
char * name = “Unknown”;
MODULE_PARM(irq,”i”); /* declare irq as int */
MODULE_PARM(name,”s”); /* declare name as string
  */
```

- Defined at load time

```
insmod mod.o irq=9 name=▪”The Server”
```

Module Info

- Externally visible module declarations used to supply clients with some useful information
- Macros

```
MODULE_AUTHOR("Somebody");
MODULE_DESCRIPTION("This module doesn't do
  anything");
MODULE_PARM_DESC(irq, "Device IRQ (3/4)"
```

struct file_operations

```
struct file_operations {
  struct module *owner;
  loff_t (*llseek) (struct file *, loff_t, int);
  ssize_t (*read) (struct file *, char *, size_t, ...
  ssize_t (*write) (struct file *, const char *, ...
  int (*readdir) (struct file *, void *, filldir_t);
  unsigned int (*poll) (struct file *, struct ...
  int (*ioctl) (struct inode *, struct file *, ...
  int (*mmap) (struct file *, struct vm_area_struct *);
  int (*open) (struct inode *, struct file *);
  int (*flush) (struct file *);
  int (*release) (struct inode *, struct file *);
  int (*fsync) (struct file *, struct dentry *, ...
  int (*fasync) (int, struct file *, int);
  int (*lock) (struct file *, int, struct file_lock *);
  :
  :
};
```

struct file

```
struct file {
  struct list_head         f_list;
  struct dentry            *f_dentry;
  struct vfsmount          *f_vfsmnt;
  struct file_operations   *f_op;
  atomic_t                 f_count;
  unsigned int             f_flags;
  mode_t                   f_mode;
  loff_t                   f_pos;
  unsigned long            f_reada, f_ramax, f_raend,
                                f_ralen, f_rawin;
  struct fown_struct       f_owner;
  unsigned int             f_uid, f_gid;
  int                      f_error;
  :
};
```

MINOR(f_dentry->d_inode->i_rdev)

Module's Reference Counter

- Automatically tracks how many clients a module has at a moment
- Avoids unloading a module that is being used by a client
- Manipulated by macros

```
MOD_INC_USE_COUNT
MOD_DEC_USE_COUNT
MOD_IN_USE
```

Programming Hits

- No standard libraries (and headers)
  - `printk` instead of `printf`

```
#include <linux/x.h>
#include <asm/y.h>
```

- Signalize kernel code

```
#define __KERNEL__
```

- Avoid name-clashes
  - Local symbols (`static`)
  - Prefixed symbols (`mod_sym`)

```
EXPORT_NO_SYMBOLS;
```

More Programming Hits

- Kernel code runs within the context of calling user process

```
#include <asm/uaccess.h>
unsigned long copy_to_user(to, from, count);
unsigned long copy_from_user(to, from, count);
```

- In-kernel memory allocation

```
#include <linux/malloc.h>
void *kmalloc(unsigned int size, int priority);
void kfree(void *obj);
```