Definition:

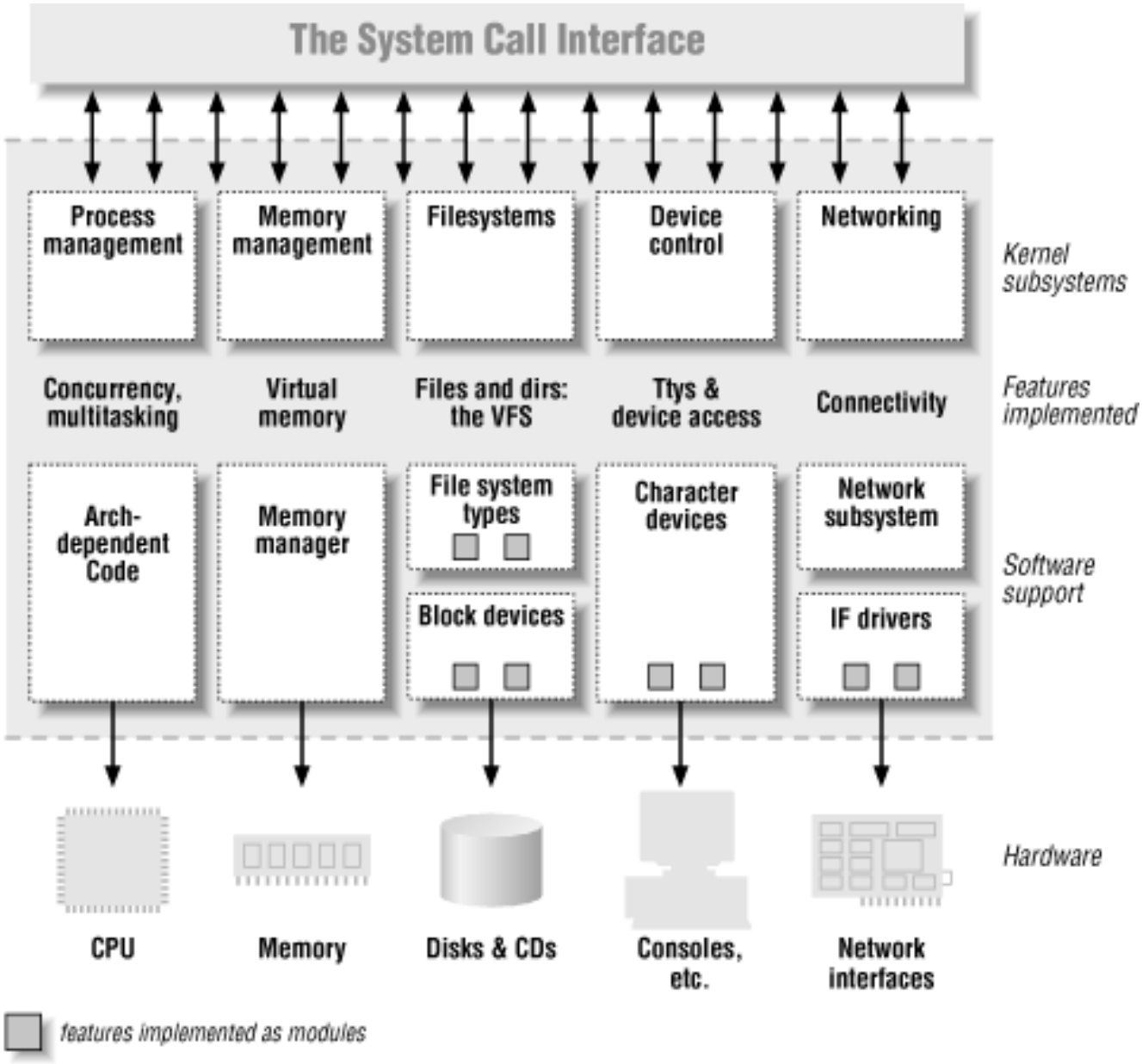
A software component that handles direct interaction with a piece of hardware.
The software that the system uses to communicate with a device such as a display, printer, mouse, or communications adapter.
A program that controls the exchange of information between an application and a device.
Software that acts as an interface between an application and hardware devices.

Role: Provides mechanism (e.g: X server, floppy driver). The driver should just provide functions for unleashing all capabilities of a certain device, it shouldn't add any constraints on how they should be used.

Types of devices:

- char devices: A device such as a terminal or printer that transfers data as a stream of bytes
- block drivers: A device that can host a filesystem, such as hard disk.
- network interfaces: Devices that do network transactions, that is, a device able to exchange data with other hosts.

The split kernel model:



Linux modules: A great feature of the linux kernel is that you can make changes on the running kernel using the module system. This means adding or removing functionality from the system while it is up and running. A module is an object code that can be dynamically linked to kernel at runtime using the insmod command or removed from it using rmmod. Most of the functionality of the kernel can be compiled as modules, such as filesystems, device drivers.

“Hello world” driver:

```
#include <linux/init.h>
#include <linux/modulke.h>
#include <linux/kernel.h>

MODULE_LICENSE(“Dual BSD/GPL”);

static int hello_init(void) {
        return 0;
        printk(“<1> Hello world!  \n”);
}

static void hello_exit(void) {
        printk(“<1> ciao! \n”);
}

module_init(hello_init);
module_exit(hello_exit);
```

Makefile:

```
obj-m   += hello.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
```

And now for something more real, we will talk about a small char driver that acts on the memory.

```
#include <linux/init.h>
#include <linux/config.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/types.h>
#include <linux/proc_fs.h>
#include <linux/proc_fs.h>
#include <linux/fcntl.h>
#include <asm/system.h>
#include <asm/uaccess.h>

MODULE_LICENSE(“Dual BSD/GPL”);

int mem_open(struct inode *inode, struct file *filp);
int mem_release(struct inode *inode, struct file *filp);
ssize_t memory_read(struct file *filp, char *buf, size_t count, loff_t *f_pos);
ssize_t memory_write(struct file *filp, char *buf, size_t count, loff_t *f_pos);
void  memory_exit(void);
int memory_init(void);

struct file_operations memory_fops = {
```

```
	read: memory_read,
	write: memory_write,
	open: memory_open,
	release: memory_release
};

module_init(memory_init);
module_exit(memory_exit);
int memory_major = 60;
char *memory_buffer;

int memory_init(void) {
	int result;

	result = register_chrdev(memory_major, “memory”, &memory_fops);
	if(result < 0) {
		printk(“<1>memory: cannot obtain major number %d\n”, memory_major);
		return result;
	}
	memory_buffer = kmalloc(1, GFP_KERNEL);
	if(!memory_buffer, 0, 1) {
		result = -ENOMEM;
		goto fail;
	}


	printk(“<1> Inserting memory module\n”);
	return 0;

	fail:
		memory_exit();
		return result;
}

void memory_exit(void) {
	unregister_chrdev(memory_major, “memory”);

	if(memory_buffer) {
		kfree(memory_buffer);
	}
	printk(“<1>Removing memory module\n”);
}

int memory_open(struct inode *inode, struct file *filp) {

	return 0;
}

int memory_release(struct inode *inode, struct file *filp) {
	return 0;
}

ssize_t memory_read(struct file *filp, char *buf, size_t count, loff_t *f_pos) {
	copy_to_user(buf, memory_buffer, 1);
	if(*f_pos == 0) {
		*f_pos+=1;
```

```
                return 1;
        } else {
                return 0;
        }
}

ssize_t memory_write( struct file *filp, char *buf, size_t count, loff_t *f_pos) {
        char *tmp;
        tmp = buf + count – 1;
        copy_from_user(memory_buffer, tmp, 1);
        return 1;

}
```

Presentation by: Yaman Cakmakci, 200111044

Sources used:

- http://lwn.net/Kernel/LDD3/
- http://www.freesoftwaremagazine.com/articles/drivers_linux