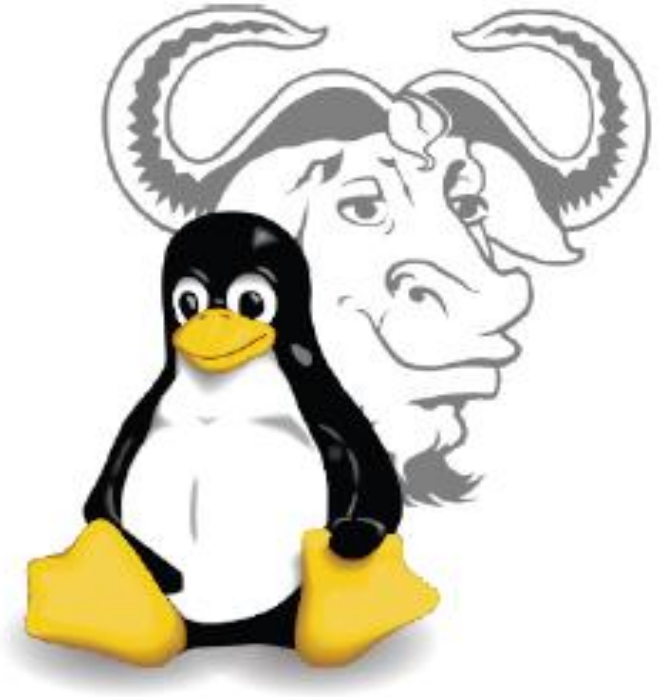
Introducción a los Device Drivers

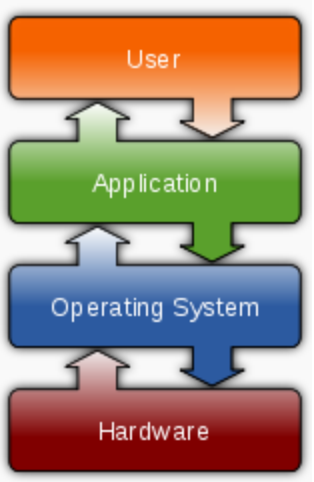
(CC) Por Daniel A. Jacoby

ITBA
INSTITUTO TECNOLOGICO DE BUENOS AIRES
•UNIVERSIDAD PRIVADA•



Que son los DD ?

- En informática un DD es un programa que permite que las aplicaciones de alto nivel interactúen con el hardware.
- Son dependientes tanto del Sistema operativo como del hardware usado.



Porque son necesarios los DD?

Un Sistema Operativo debe poder evolucionar

- Incorporación de nuevos dispositivos
- Flexibilidad para la corrección de errores
- Permitir la optimización del funcionamiento

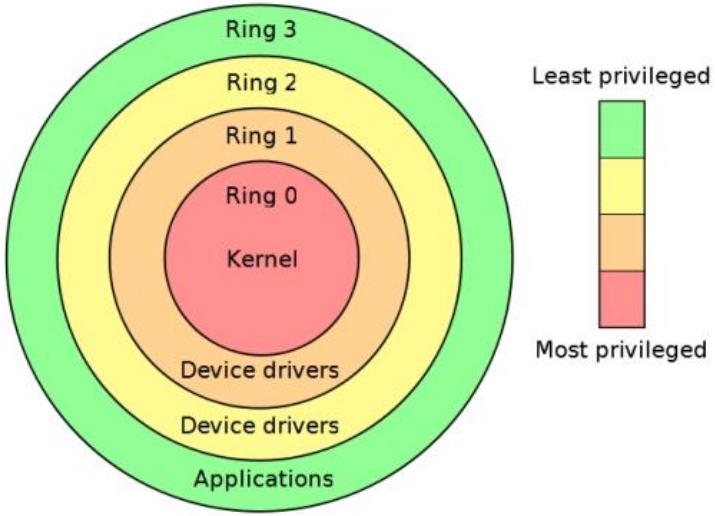
Como ???

LKM

- Programación de código abierto (Open Source)
- LKM (Loadable Kernel Modules)

Acceso al Hardware

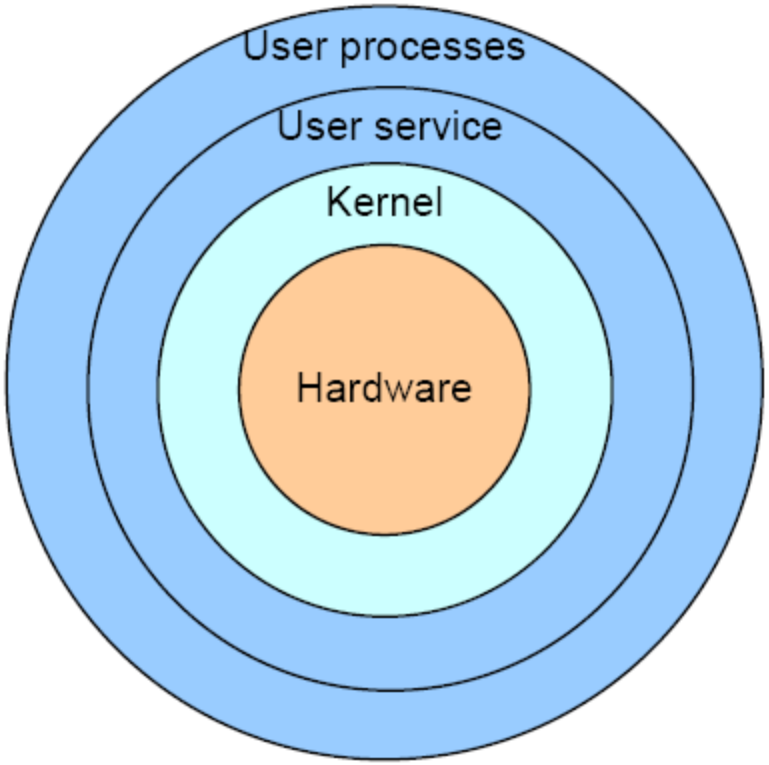
- Protección del OS y Aplicaciones !!!
- Windows & Linux → supervisor/user-mode



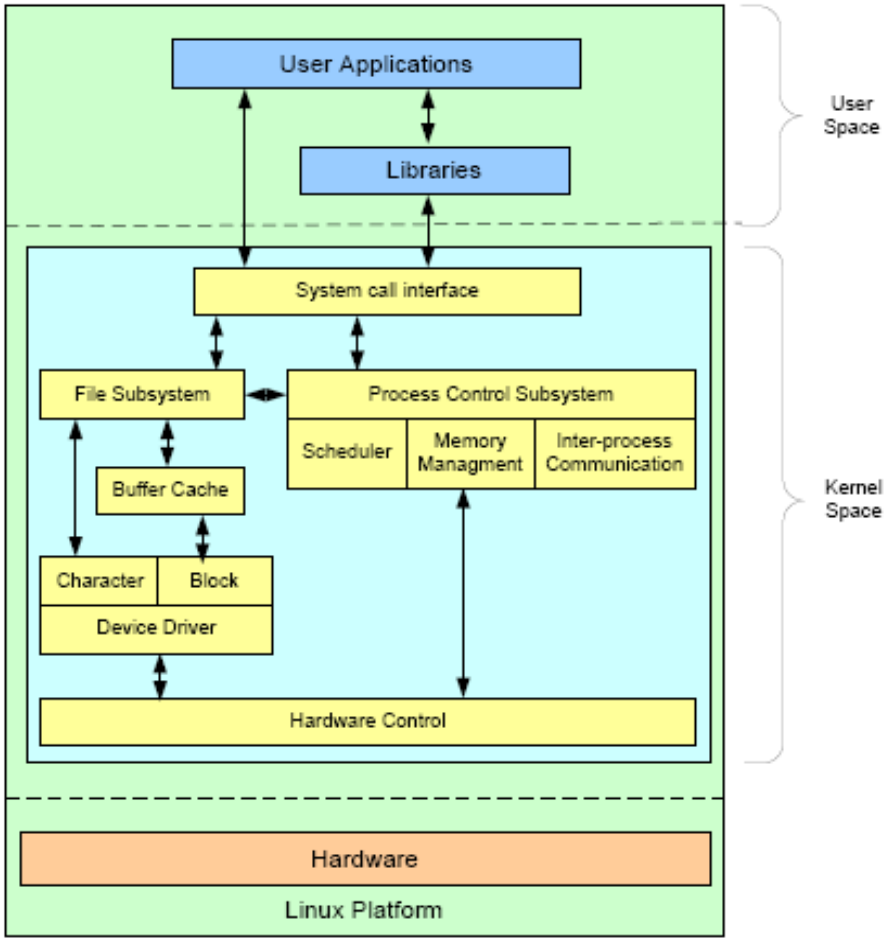
Kernel y User space

- **Kernel Space** : Acceso directo al hardware de manera organizada. Impedir que el usuario acceda a recursos del hardware de cualquier forma
- **User space**: Aplicaciones del usuario que deberán estar controladas para evitar hacer daño al Sistema operativo u otras aplicaciones (Ring3)

Interfaz US-KS



Interfaz US-KS



Interfaz básica de un LKM

Event	User function	Kernel function
Load module	insmod	module_init()
Open device	fopen	file_operations:open
Read device	fread	file_operations:read
Write device	fwrite	file_operations:write
Close device	fclose	file_operations:release
Remove device	rmmod	module_exit()

Interfaz mínima

- int **init_module**(void);

  Es invocada durante la instalación del modulo

- void **cleanup_module**(void);

  Es invocada durante la remoción del modulo

Interfaz mínima (usando macros)

- **module_init** (x);

  Es invocada durante la instalación del modulo

- **module_exit**(x);

  Es invocada durante la remoción del modulo

PRINTK

La funcion printk() es usada en la programacion de device drivers para enviar mensajes al log del kernel

Sintaxis :

printk ("log level" "message", <arguments>);

```
#define KERN_EMERG "<0>" /* system is unusable*/
#define KERN_ALERT "<1>" /* action must be taken immediately*/
#define KERN_CRIT "<2>" /* critical conditions*/
#define KERN_ERR "<3>" /* error conditions*/
#define KERN_WARNING "<4>" /* warning conditions*/
#define KERN_NOTICE "<5>" /* normal but significant condition*/
#define KERN_INFO "<6>" /* informational*/
#define KERN_DEBUG "<7>" /* debug-level messages*/
```

Ejemplo

```
#include <linux/module.h>

int init_module(void)
{ printk("<1>Hello, world\n"); return 0; }

void cleanup_module(void)
{ printk("<1>Goodbye cruel world\n"); }
```

Ejemplo

```
/*
 * hello.c  Hello, World! As a Kernel Module
 */

#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

/*
 * hello_init  the init function, called when the module is loaded.
 * Returns zero if successfully loaded, nonzero otherwise.
 */
static int hello_init(void)
{
        printk(KERN_ALERT "I bear a charmed life.\n");
        return 0;
}

/*
 * hello_exit  the exit function, called when the module is removed.
 */
static void hello_exit(void)
{
        printk(KERN_ALERT "Out, out, brief candle!\n");
}

module_init(hello_init);
module_exit(hello_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Shakespeare");
```

Compilacion: Makefile

```
ifneq   ($(KERNELRELEASE),)
obj-m   := hello.o

else
KDIR    := /lib/modules/$(shell uname -r)/build
PWD     := $(shell pwd)
default:
        $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
        rm -r .tmp_versions *.mod.c .*.cmd *.o

endif
```

Compilacion

- Make File : mmake

./mmake mydriver

mydriver.c → mydriver.ko

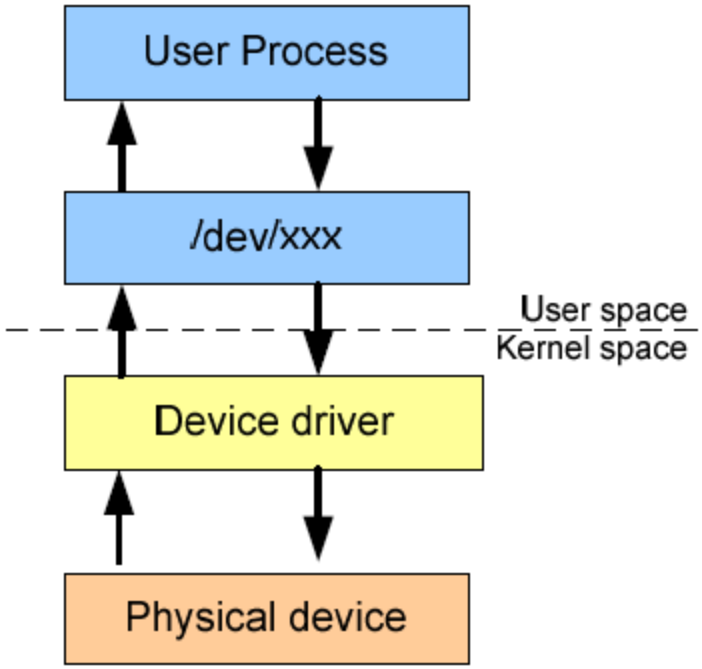
Instalacion/Remocion

- sudo insmod mydriver.ko
- sudo rmmod mydriver

Device Nodes

```
# ls -l /dev/lp*

crw-rw-rw 1 root root 6, 0 April 23 1994 /dev/lp0
```



comandos utiles

- lsmod
- cat /proc/devices
- cat /proc/ioports