Seguridad Informática

Lenguaje C y llamadas al Sistema.

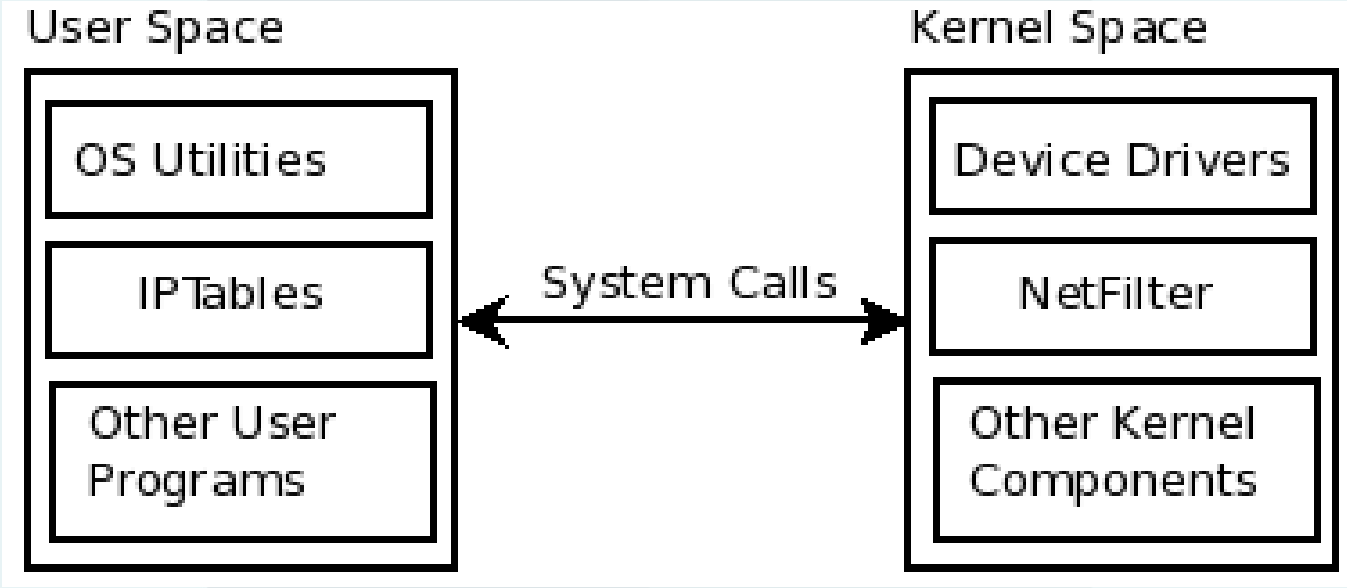
Fernando C.

Temario.

- **Interacción de C con el sistema operativo.**
- **Clasificación de llamadas al sistema.**
- **Implementación de llamadas al sistema.**
- Módulos de kernel.
- “Programación segura”.

Interaccion con el sistema operativo.

- Los sistemas operativos tienen una caracteristica peculiar, y que los hace ser “seguros” a la hora de ejecutar un proceso, espacio de kernel y espacio de usuario.



¿Qué es un kernel o nucleo?

“El corazón de este sistema operativo”

El programa principal encargado de:

- Administración de la memoria para todos los programas y procesos en ejecución.
- Administración del tiempo de procesador que los programas y procesos en ejecucion utilizan.
- Es el encargado de que podamos acceder a los periféricos/elementos de nuestra PC. “/dev/”

Llamadas al sistema.

- man syscalls

Clasificacion de las llamadas al sistema.

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

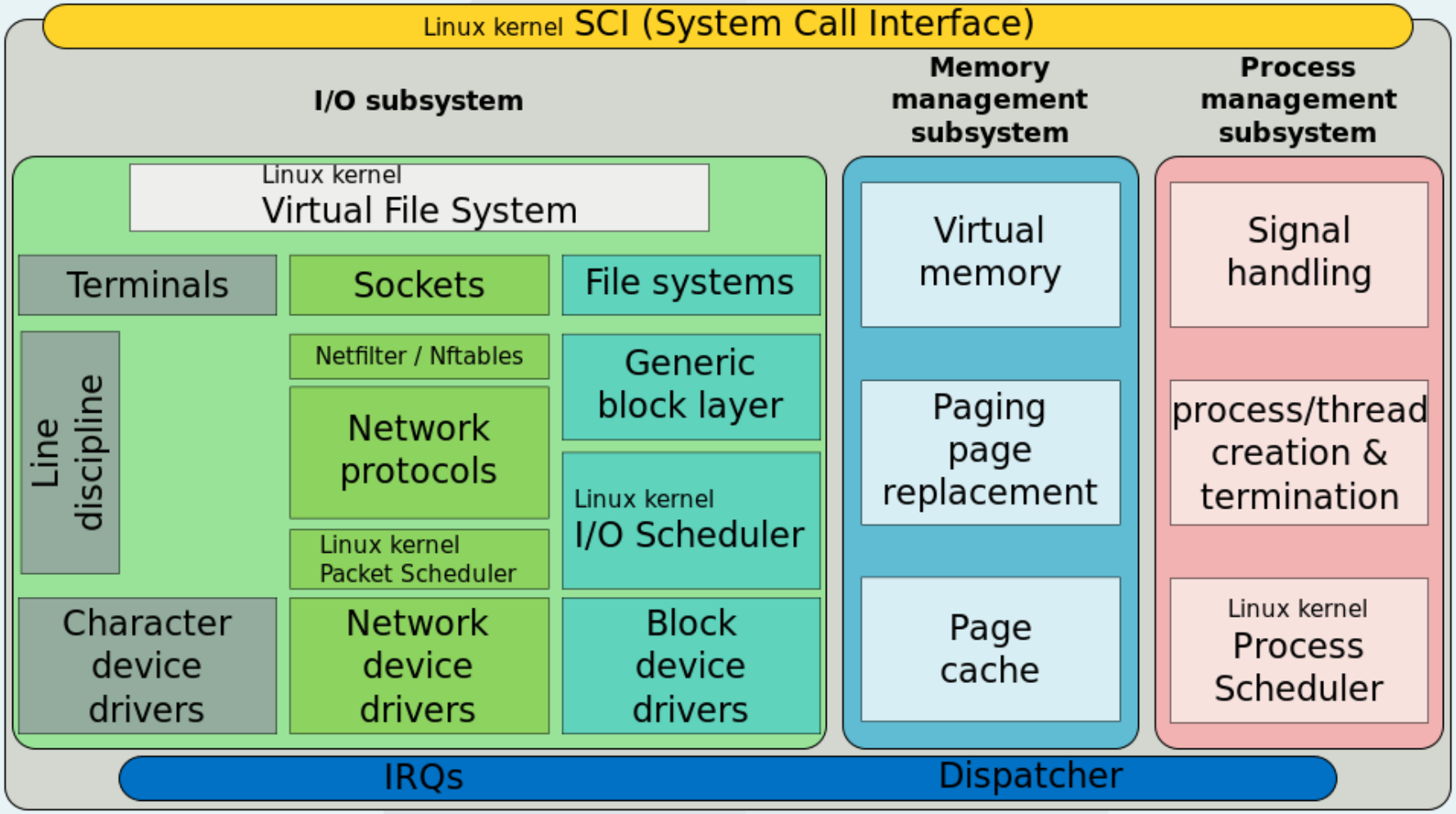
Como funciona una llamada al sistema?.

Cuando una llamada al sistema ocurre, se le envia una interrupcion a la aplicacion que la genera y se le cede el control al kernel.

Si vemos a nivel de lenguaje ensamblador, la instruccion 0x80, genera la interrupcion para hacer la llamada al sistema.

“strace”

Devices.


Linux kernel SCI (System Call Interface)
I/O subsystem
Memory management subsystem
Process management subsystem
Linux kernel
Virtual File System
Terminals
Sockets
File systems
Line discipline
Netfilter / Nftables
Network protocols
Linux kernel
Packet Scheduler
Generic block layer
Linux kernel
I/O Scheduler
Character device drivers
Network device drivers
Block device drivers
Virtual memory
Paging page replacement
Page cache
Signal handling
process/thread creation & termination
Linux kernel
Process Scheduler
IRQs
Dispatcher

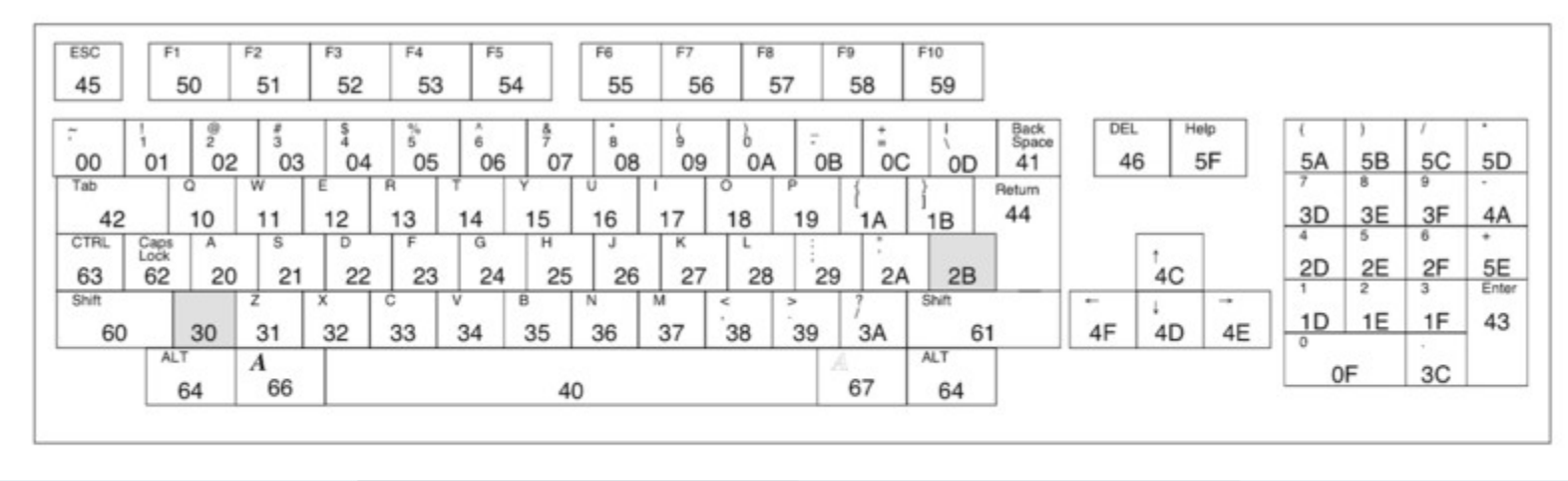
Devices.

- /dev/
- /dev/sda
  - /dev/sda1
  - /dev/sda2
- /dev/sdb
  - /dev/sdb1
- /dev/tty
- /dev/random
- /dev/zero
- /dev/null

Input devices.

- /dev/input/event0

- Como lidia el sistema operativo con entradas del teclado, como las convierte a lo que vemos como caracteres?

- 

Input devices.

/usr/include/linux/input.h

- Notese la estructura: input_event

```
struct input_event ev;
read(fd, &ev, sizeof(struct input_event));
```

Practica 18. Keylogger.

Si recuerdas como abrir archivos con descriptores, esa, es la forma de programacion mas “cercana al kernel” podemos tener en C, felicidades, ya la has usado, puedes programar ya tu sistema operativo.

Crea un programa que lea el dispositivo de entrada correspondiente a tu teclado, tendras generados valores numericos que corresponden a cada una de las entradas, pero yo quiero leer caracteres, no numeros, ya tienes el mapa de caracteres, hazlo!

Imprime en pantalla lo que leas de este dispositivo.

Pista, hay un header que vimos en la diapositiva pasada, incluyelo, al igual que este sys/types.h.

Practica 19. Lenght.

Realiza un programa con la llamada al sistema lseek, que me devuelva el tamanio en bits, bytes y kilobytes de un archivo, quiero la ubicacion del archivo por linea de comandos.

Pista. Ver las diapositivas de ayer.