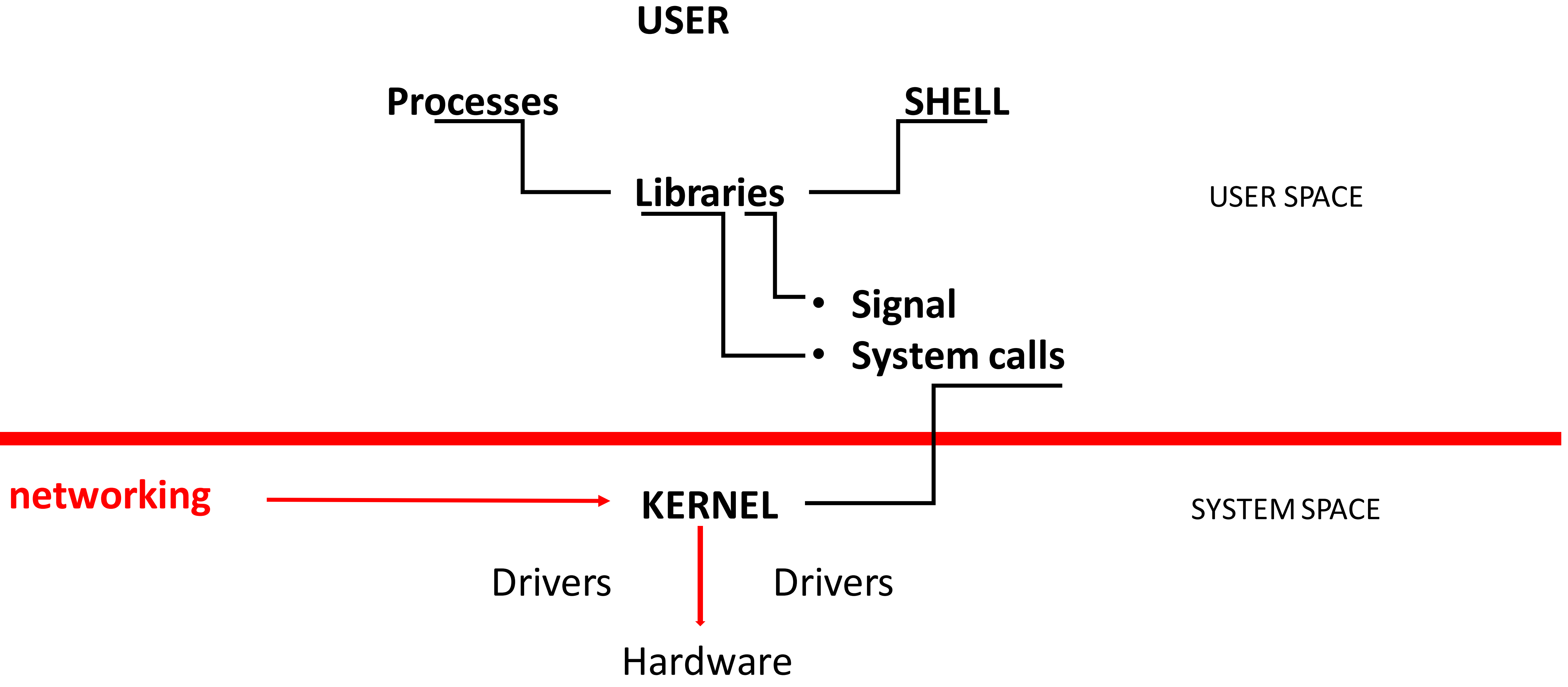
How Linux is organized

- Understanding the stack
- Understanding the role of the kernel
- Understanding drivers, kernel modules and device files
- Understanding glib
- Understanding the Linux shell
- Understanding file descriptors

Understanding the stack



USER

Processes

SHELL

Libraries

USER SPACE

- Signal
- System calls

networking

KERNEL

SYSTEM SPACE

Drivers

Drivers

Hardware

Linux Kernel

The Linux® kernel is the main component of a Linux operating system (OS) and is the core interface between a computer's hardware and its processes. It communicates between the 2, managing resources as efficiently as possible.

The kernel is so named because—like a seed inside a hard shell—it exists within the OS and controls all the major functions of the hardware, whether it's a phone, laptop, server, or any other kind of computer.

What the kernel does

The kernel has 4 jobs:

- **Memory management:** Keep track of how much memory is used to store what, and where
- **Process management:** Determine which processes can use the central processing unit (CPU), when, and for how long
- **Device drivers:** Act as mediator/interpreter between the hardware and processes
- **System calls and security:** Receive requests for service from the processes

Where the kernel fits within the OS

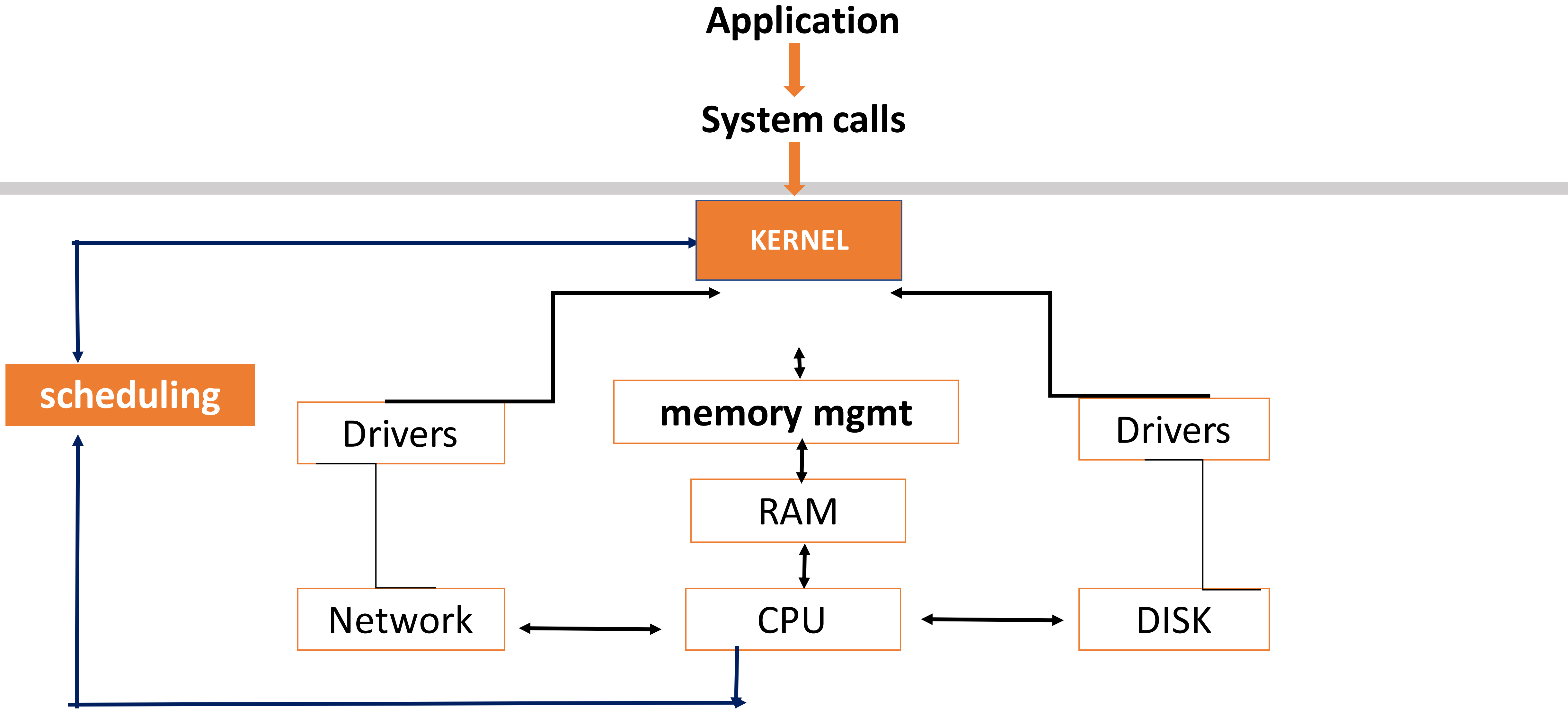
To put the kernel in context, you can think of a Linux machine as having 3 layers:

The hardware: The physical machine—the bottom or base of the system, made up of memory (RAM) and the processor or central processing unit (CPU), as well as input/output (I/O) devices such as storage, networking, and graphics. The CPU performs computations and reads from, and writes to, memory.

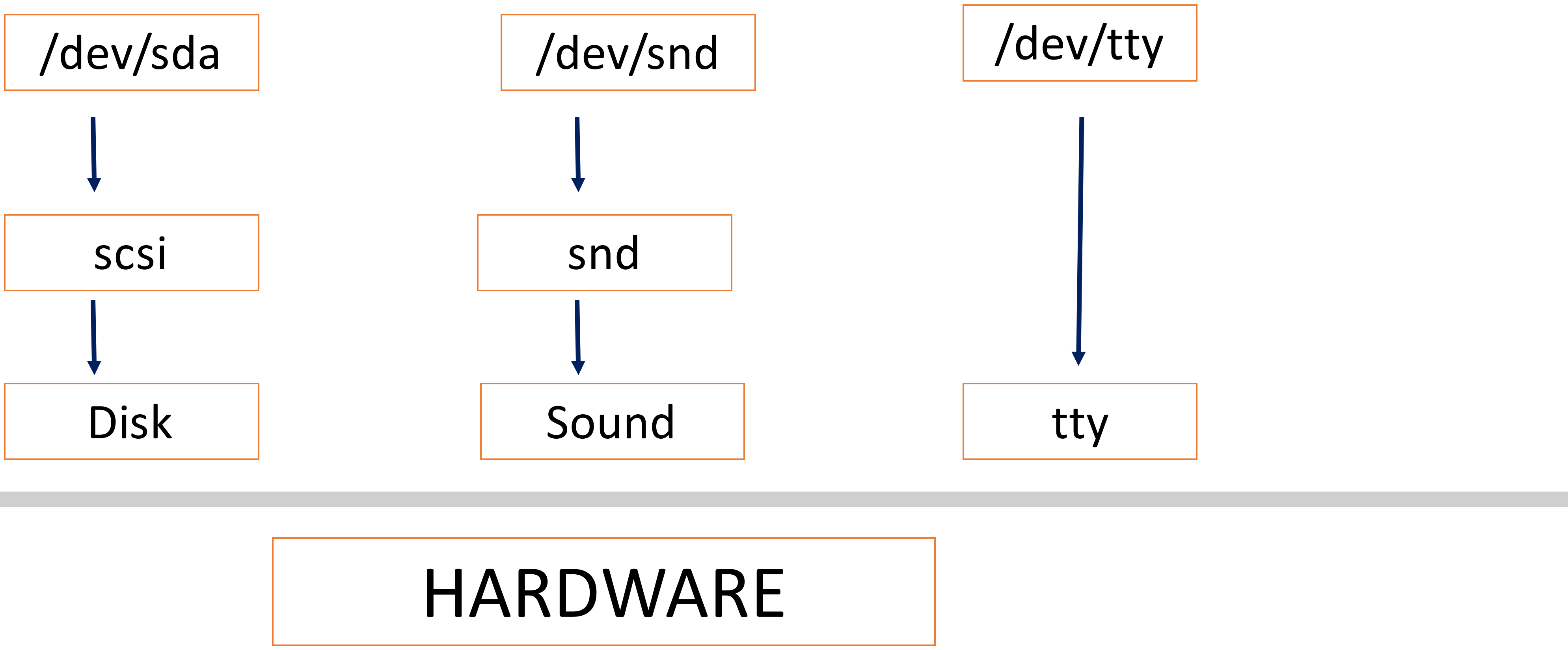
The Linux kernel: The core of the OS. (See? It's right in the middle.) It's software residing in memory that tells the CPU what to do.

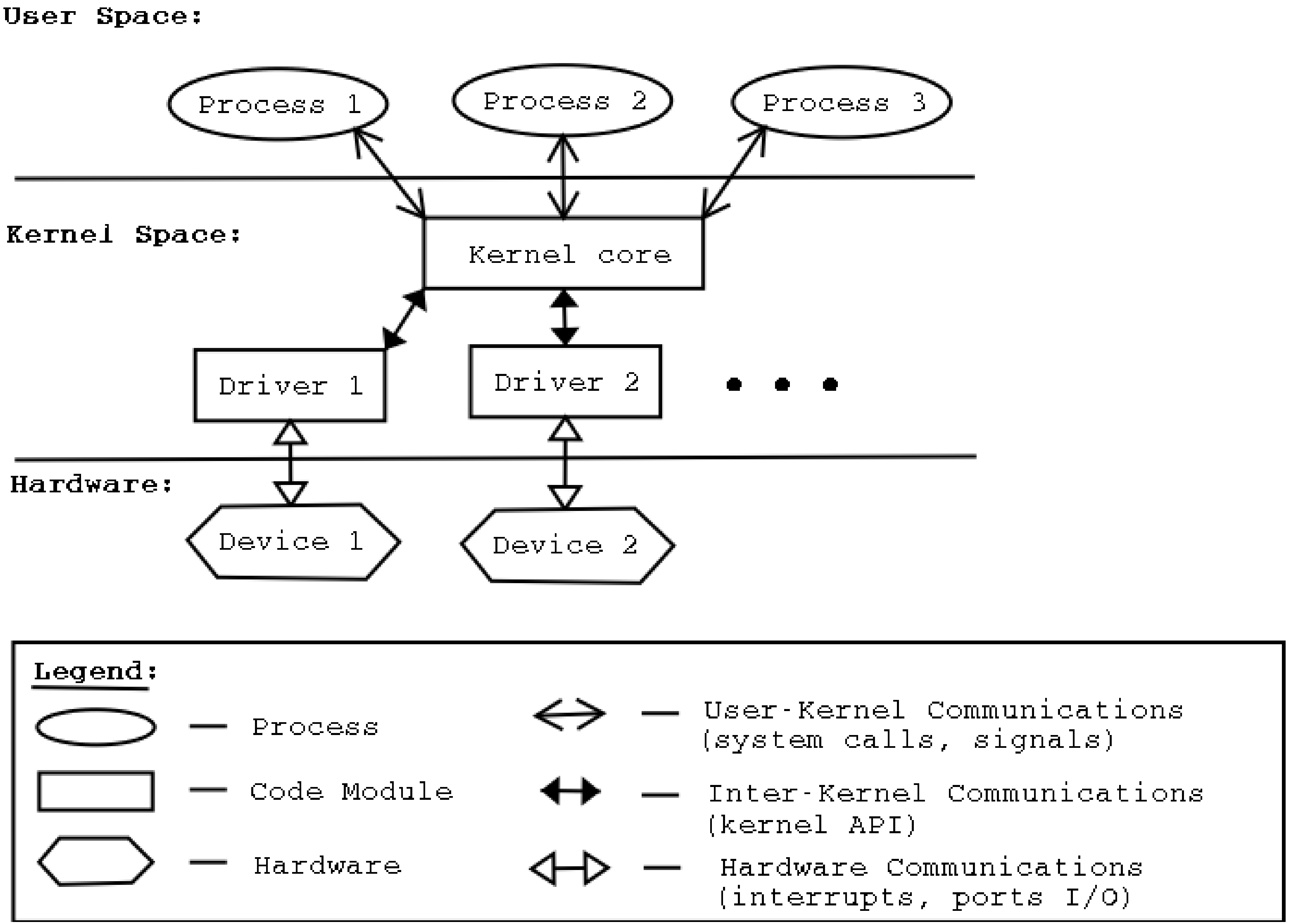
User processes: These are the running programs that the kernel manages. User processes are what collectively make up user space. User processes are also known as just *processes*. The kernel also allows these processes and servers to communicate with each other (known as inter-process communication, or IPC).

Understanding the role of the kernel



Understanding drivers, kernel modules & device files




User Space:
Process 1
Process 2
Process 3
Kernel Space:
Kernel core
Driver 1
Driver 2
• • •
Hardware:
Device 1
Device 2
Legend:
— Process
— Code Module
— Hardware
— User-Kernel Communications (system calls, signals)
— Inter-Kernel Communications (kernel API)
— Hardware Communications (interrupts, ports I/O)

Show the status of modules in the Linux Kernel

- #lsmod

Show information about a Linux Kernel module

- #modinfo modulename

Add and remove modules from the Linux Kernel

- #modprobe modulename

Show information about hardware devices

- #ls –l /dev/sda
- #ls –l /dev/tty

Understanding glibc

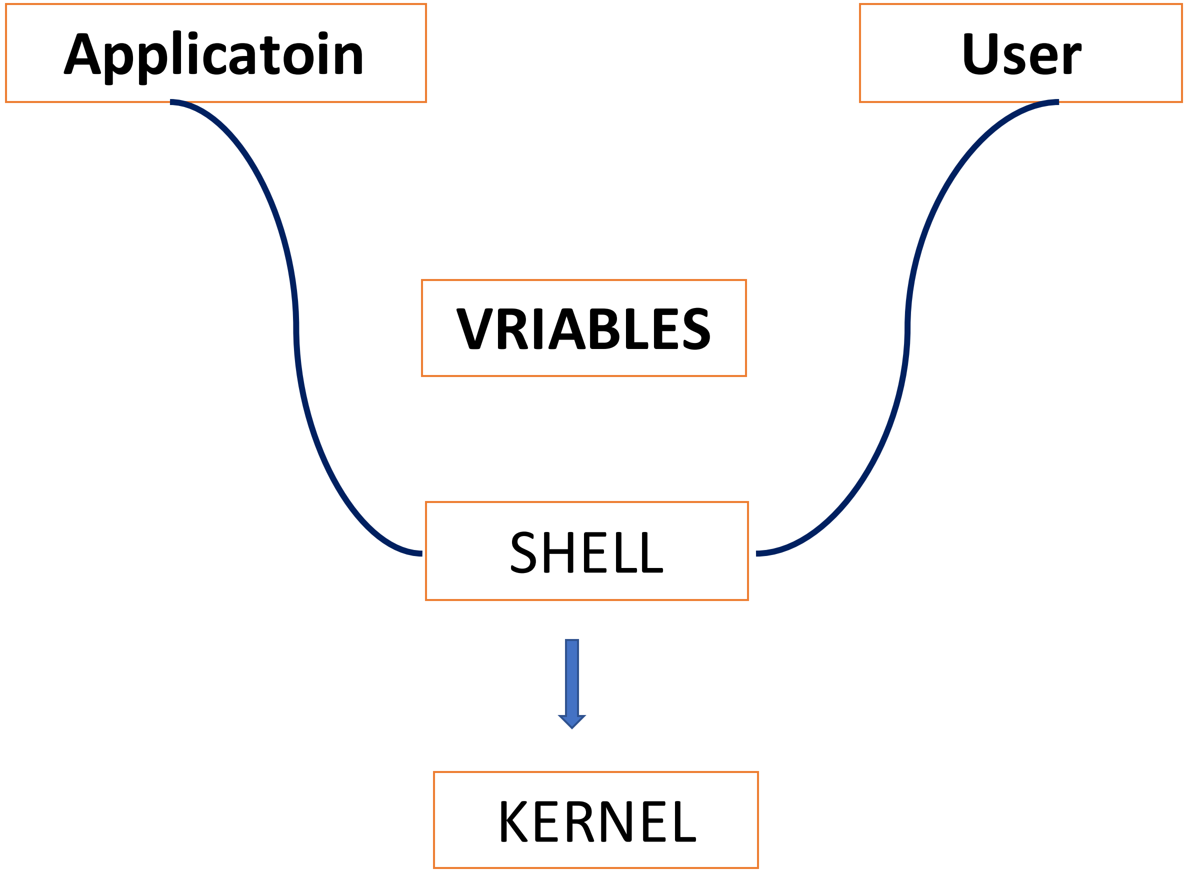
- The term "libc" is commonly used as a shorthand for the "standard C library", a library of standard functions that can be used by all C programs
- Linux is written in the C Programming language
- Most Linux Components are written in C
- C is Considered a low level-language
- High-Level Languages such as scripting languages as well as are common as well
- Python is an often used scripting language, which itself is written in C

#ldd print shared object dependencies

- #ldd $(which ls)

- #ldd /usr/sbin/fdisk

Understanding the Linux shell


Applicatoin

User

VRIABLES

SHELL

KERNEL

We can see the environment file which is right here

- #cd /usr/lib/systemd/system
- #grep - - sysconfig *

Understanding file descriptors

- Linux is a file oriented operating system
- Everything is happening as a file: device access, I/O handling, inter process communication (IPC) and more
- Every process keeps a table of file descriptor that shows files that are currently in use
- Common file descriptors are:
  - 0: STDIN
  - 1: STDOUT
  - 2: STDERR

We can see the file descriptors values

```
# cd /proc

#ls

#cd PID

#ls

#cat cmdline

#cd fd

#ls -l
```

Git Clone Linux Kernel

```
#git clone https://github.com/coreutils/coreutils.git
```