System and Network Administration

udev

openVPN

- A **tun** interface is a device driver that that looks like point-to-point network hardware to the operating system
- A **tap** interface is similar, but it emulates ethernet rather than point-to-point.
- Rather being connected to a wire, the driver connects to "user space", where a program can open the device just like a file and read and write packets from and to it.
- The VPN essentially links a local tun or tap device with a remote one of the same type.
- packets are encrypted and encapsulated in UDP

SO …

- Where do tun/tap devices come from?
- How do I know what modules are loaded into the kernel?

We need to understand

- device drivers
- modprobe, lsmod
- udev, /dev/ and /sys/

Device Drivers

Accessed through special files in the /dev directory

- Major device number
- Minor device number (unit)

udev - Autosense devices

- udevd
- /etc/udev/udev.conf

Device files created with

```
mknod name type major minor
```

or with

```
    cd /dev
    ./MAKEDEV pty
```

Kernel Modules

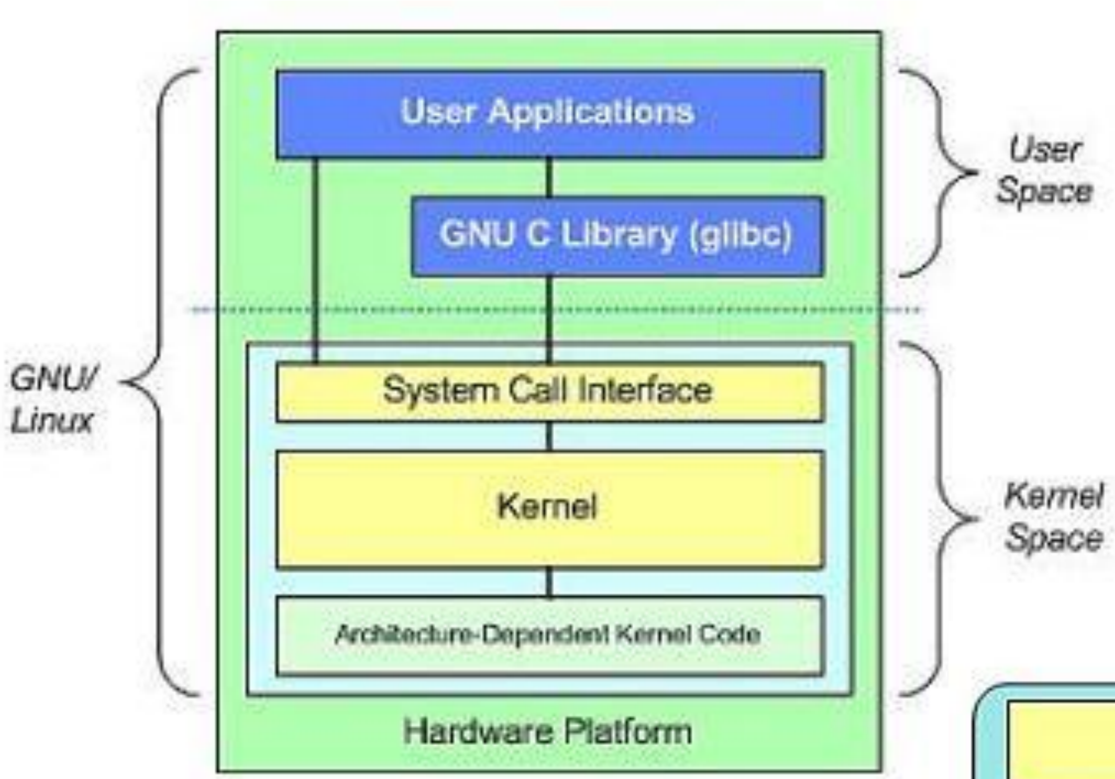
Added or removed while kernel is running

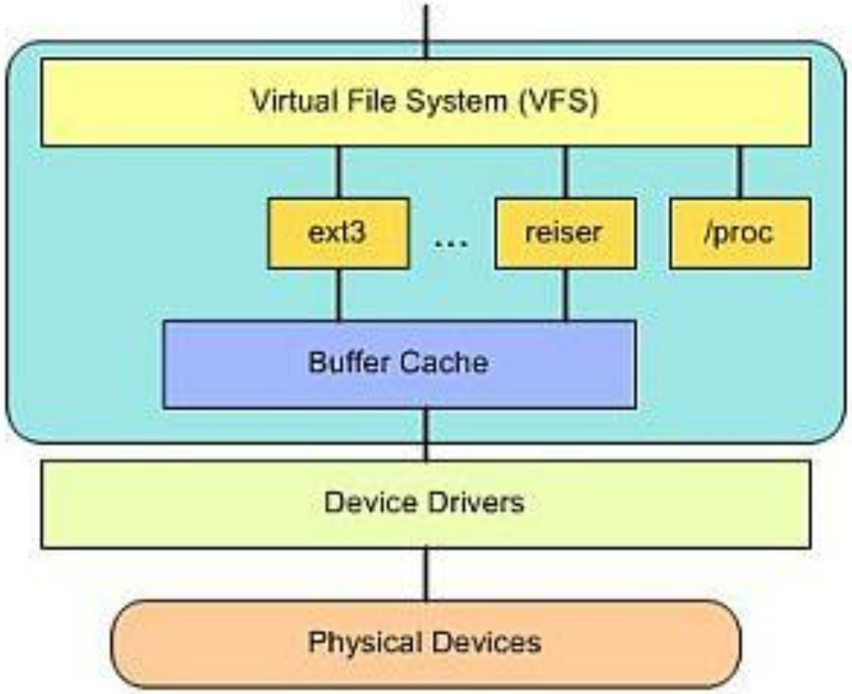
Commands:

```
    lsmod
    insmod
    rmmod
    modprobe
```

Kernel modules and Device Drivers

- Every device in the system is represented by a file - even to read from memory is to read from a file.
- In kernel space, the virtual file system (VFS) decodes the file type and transfers the file operations to the appropriate channel, like a filesystem module in case of a regular file or directory, or a device driver in case of a device file.
- All of the functions used to access a specific device are jointly referred to as the device driver.
- Device drivers must be statically compiled into the kernel or stored on disk as a dynamically loadable module.


User Applications
GNU C Library (glibc)
User Space
GNU/ Linux
System Call Interface
Kernel
Kernel Space
Architecture-Dependent Kernel Code
Hardware Platform


Virtual File System (VFS)
ext3
...
reiser
/proc
Buffer Cache
Device Drivers
Physical Devices

Device Nodes

- Device files (also called device nodes) are located in the */dev/* directory.
- A device node is a file with type c (for "character" devices, devices that bypass the buffer cache) or b (for "block" devices, which go through the buffer cache).
- Each device file is assigned a major number and a minor number.
- Traditionally, each device driver has a major device number, and all device files for devices controlled by that driver have the same major number.

Device Files and udev

```
# ls -l /dev/sda
brw-rw---- 1 root disk 8, 0 March 9 07:56 /dev/sda
```

Shows type (b=block) and permissions (rw-rw----),
    owner (root), group (disk),
    major device number (8), minor device number (0),
    date (March 9), hour (07:56) and device name (guess :-)

Modern Linux kernels allow multiple drivers to share major numbers, but most devices that you will see are still organized on the one-major-one-driver principle.

The kernel will assign a major/minor number pair when it detects a hardware device.

The kernel and udev

- The udev daemon listens to the netlink socket that the kernel uses for communicating with user space applications.
- For every detected device, the kernel creates an internal device structure (in “kernel space”) and the driver core sends an event to udev (in “user space”)
- If a device driver needs to be loaded, udev calls modprobe with parameters provided by the kernel

/dev/ and /sys/

- Once the proper device driver is loaded, udev processes the event (see next) and creates the device node in */dev/*
- Then, for every device the kernel has detected and initialized, a directory with the device name is created that contains files with device specific properties in */sys/*

udev rules

- Every event is matched against the set of rules provided in */etc/udev/rules.d/*.rules*.
- These rules can add or change event environment keys, request a specific name for the device node to be created, add symlinks pointing to the node or add programs to be run after the device node is created.
- udev rules can match on any property the kernel exports in /sys/ or adds to the event, and the rule may also request additional information from external programs

udev

- From the user space perspective, there is no difference between a device coldplug sequence and device discovery during runtime (hotplug).
- All devices that are plugged in or removed will cause an uevent to be sent to the udev daemon, which runs an event process to match against udev rules, create/remove the device node and symlinks as required, and possibly run specified programs to set up/clean up after the device.

modprobe

- `modprobe` is used to load and unload kernel modules
- `lsmod` is used to list currently loaded modules
- If you don’t know why a module is needed, use `modinfo` to find information about it

There is a fuller explanation on the website

Q&A