Linux Kernel Architecture

Amir Hossein Payberah

payberah@yahoo.com

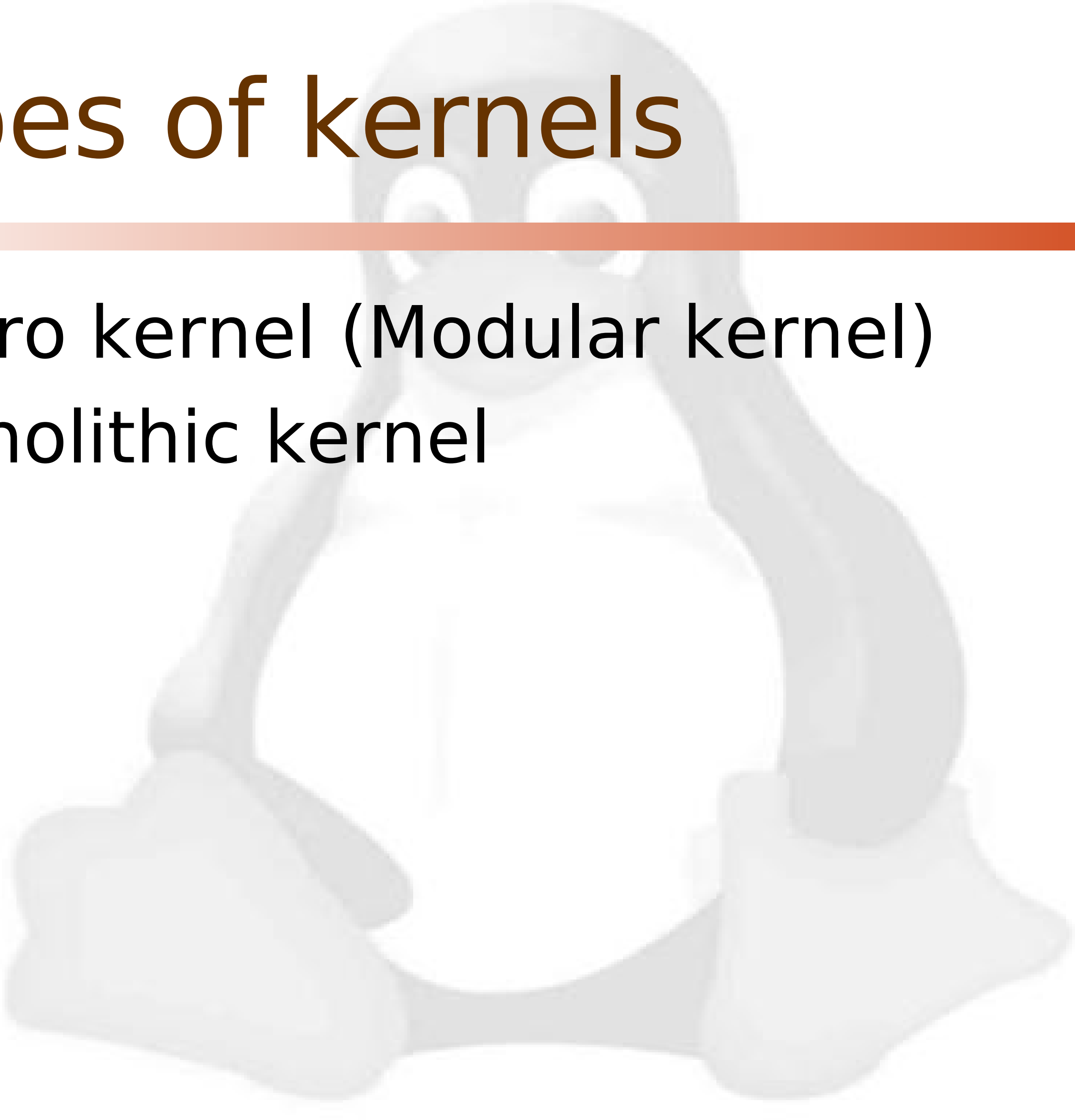
Contents

- What is Kernel ?
- Kernel Architecture Overview
  - □ User Space
  - □ Kernel Space
- Kernel Functional Overview
  - □ File System
  - □ Process Management
  - □ Device Driver
  - □ Memory Management
  - □ Networking

What is Kernel ?

- Modules or sub-systems that provide the operating system functions.
- The Core of OS

Types of kernels



- Micro kernel (Modular kernel)
- Monolithic kernel

Micro kernel

- It includes code only necessary to allow the system to provide major functionality.
  - IPC
  - Some memory management
  - Low level process management & scheduling
  - Low level input / output
- Such as Amoeba, Mach and …

Monolithic kernel

- It includes all the necessary functions.
- Such as Linux and …

Micro vs. Monolithic

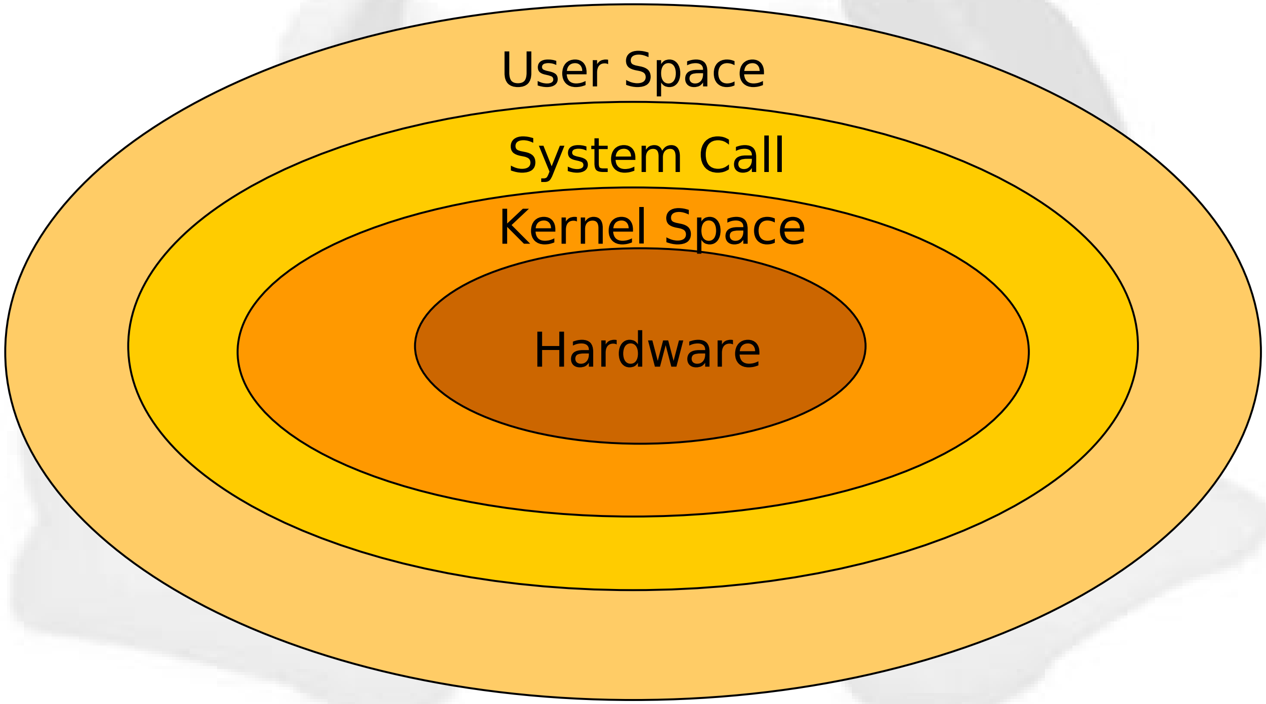
- Micro
  - Flexible
  - Modular
  - Easy to implement
- Monolithic
  - Performance

Contents

- What is Kernel ?
- Kernel Architecture Overview
  - □ User Space
  - □ Kernel Space
- Kernel Functional Overview
  - □ File System
  - □ Process Management
  - □ Device Driver
  - □ Memory Management
  - □ Networking

Kernel Architecture Overview

- User Space
- Kernel Space



User Space

- The User Space is the space in memory where user processes run.
- This Space is protected.
  - □ The system prevents one process from interfering with another process.
  - □ Only Kernel processes can access a user process

Kernel Space

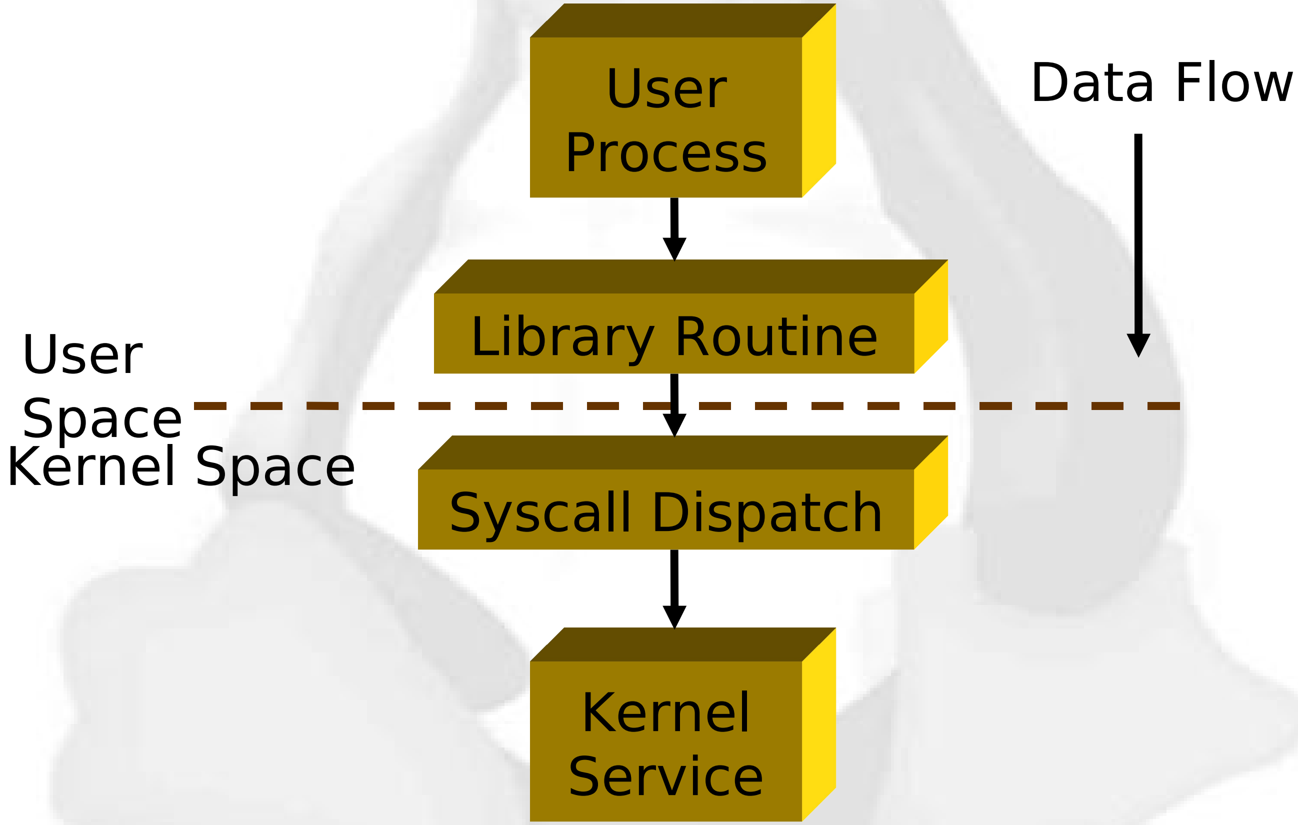
- The kernel Space is the space in memory where kernel processes run.
- The user has access to it only through the system call.

System Call

- User Space and Kernel Space are in different spaces.
- When a System Call is executed, the arguments to the call are passed from

  User Space to Kernel Space.
- A user process becomes a kernel process when it executes a system call.

User Space and Kernel Space Relationship



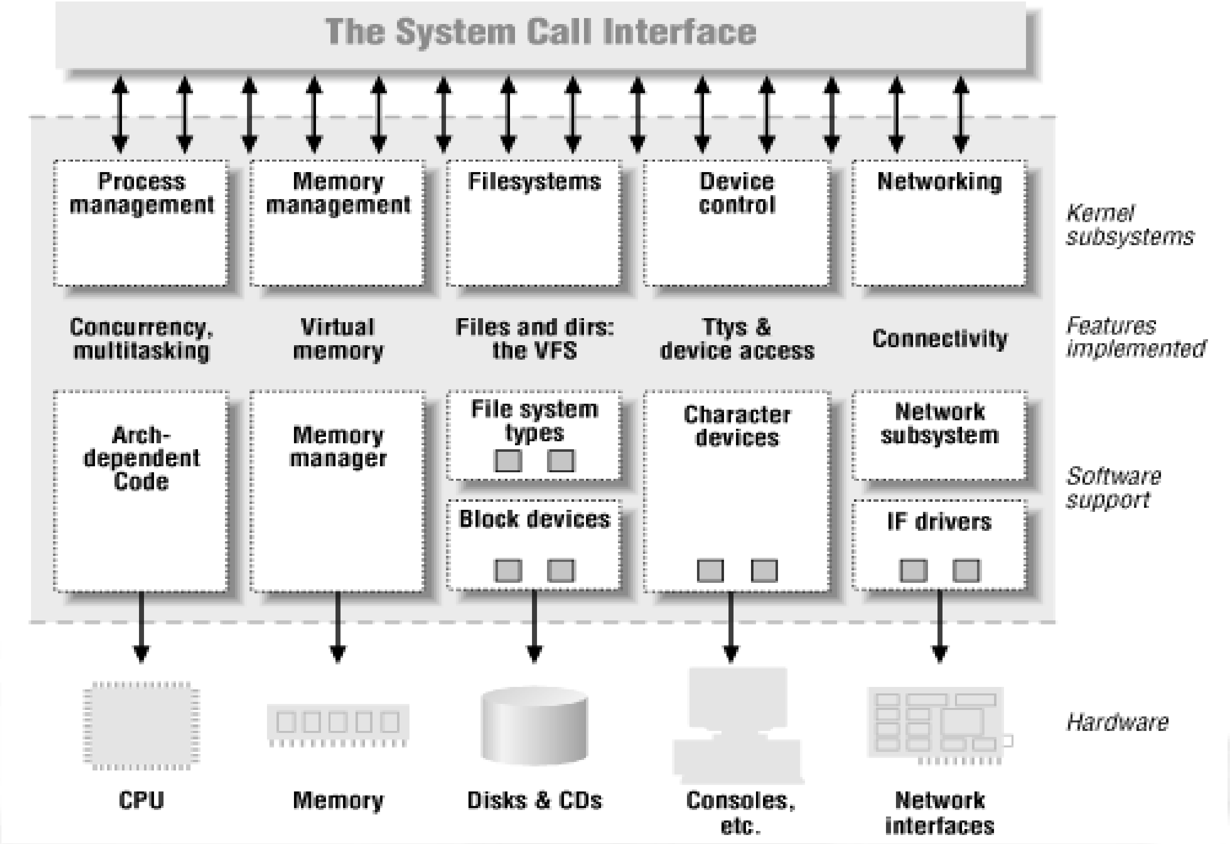
Contents

- What is Kernel ?
- Kernel Architecture Overview
  - User Space
  - Kernel Space
- Kernel Functional Overview
  - File System
  - Process Management
  - Device Driver
  - Memory Management
  - Networking

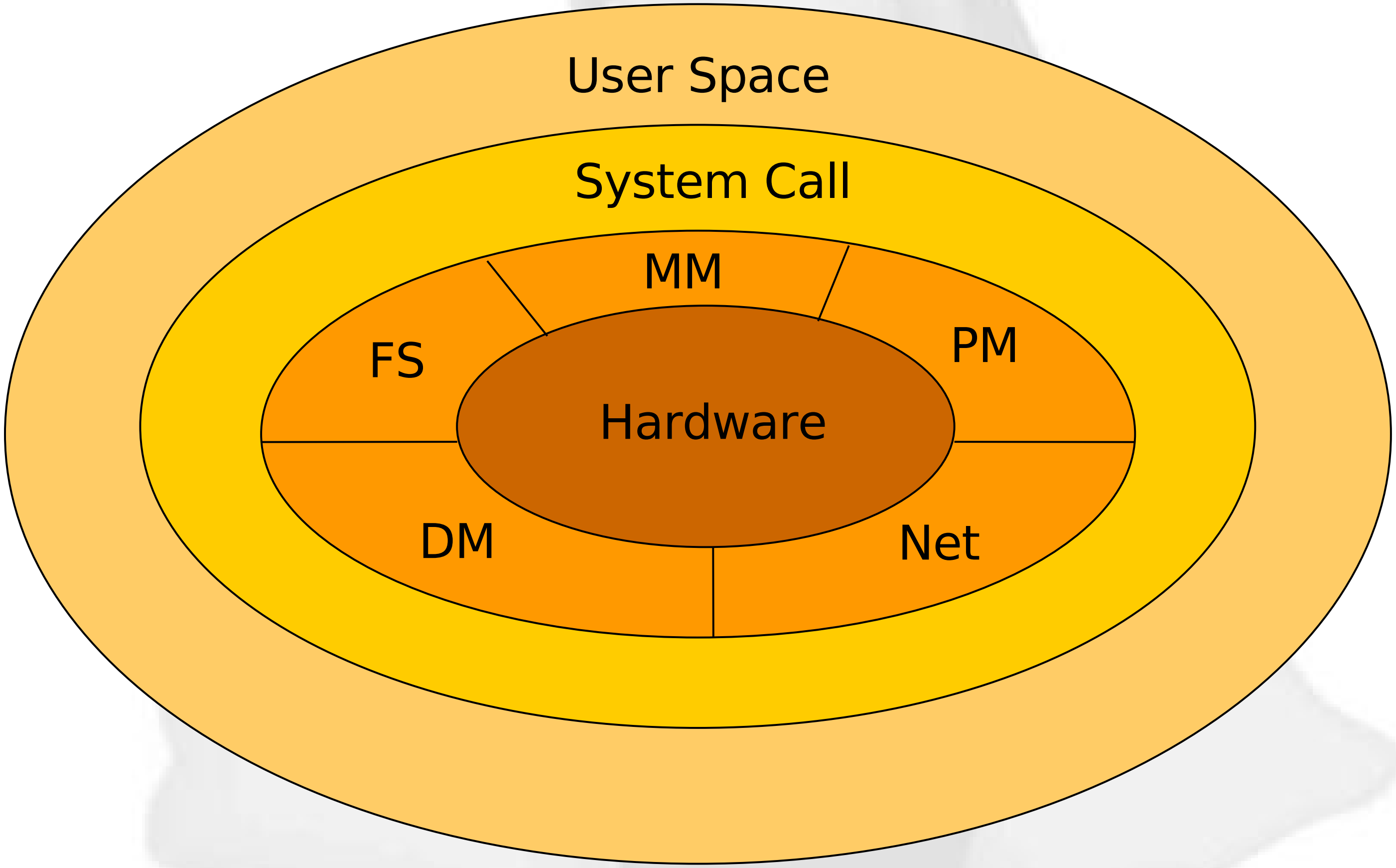
Kernel Functional Overview

- File System
- Process Management
- Device Driver
- Memory Management
- Networking

Kernel Functional Overview



Functional Layer & Architectural Layer



Contents

- What is Kernel ?
- Kernel Architecture Overview
  - □ User Space
  - □ Kernel Space
- Kernel Functional Overview
  - □ File System
  - □ Process Management
  - □ Device Driver
  - □ Memory Management
  - □ Networking

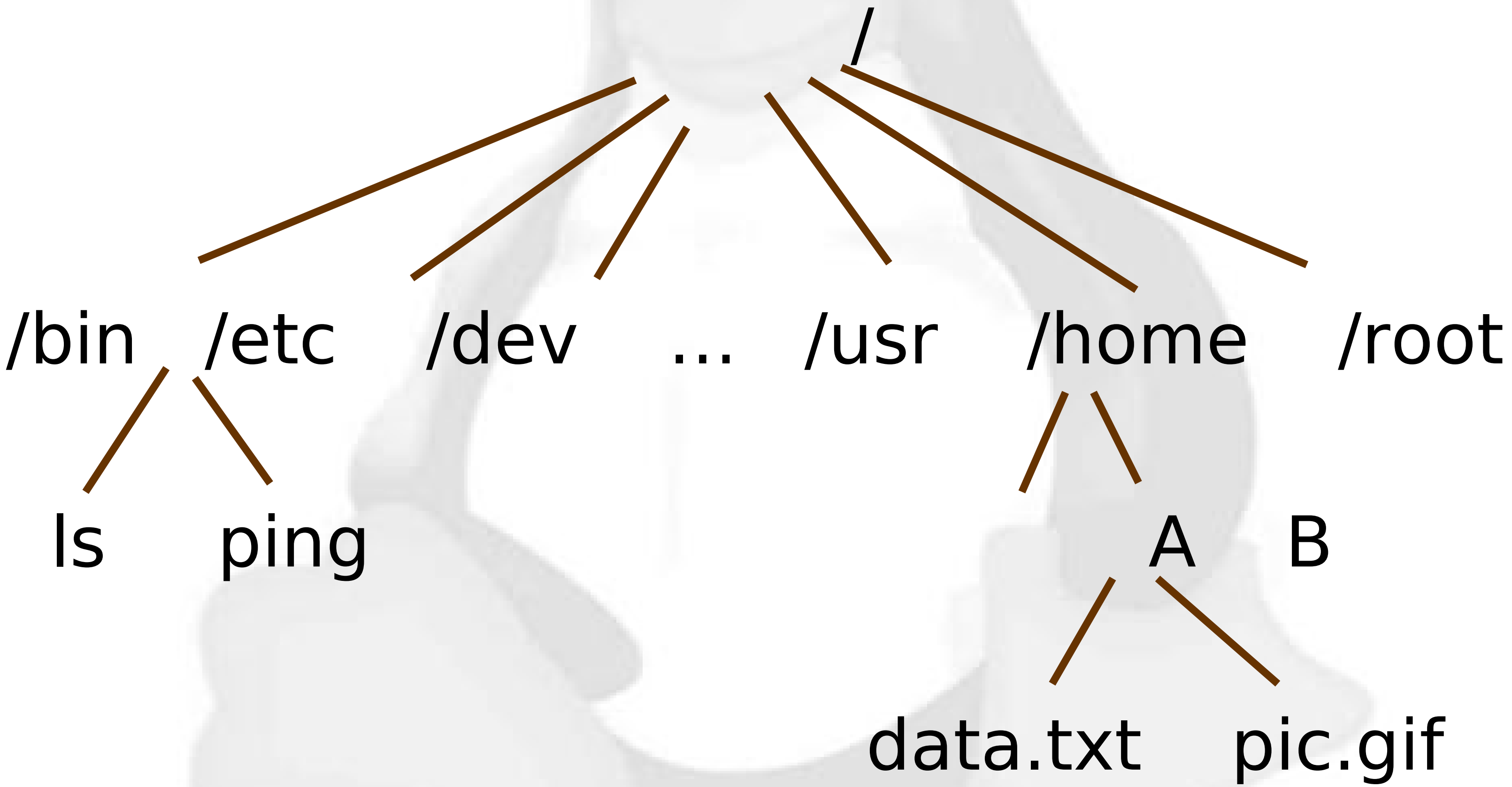
File System

- It is responsible for storing information on disk and retrieving and updating this information.
- The File System is accessed through system calls such as : open, read, write, ...
- Example :
  - □ FAT16, FAT32, NTFS
  - □ ext2, ext3
  - □ ...

Type of Files

- The Unix system has the following types of files:
  - □ Ordinary Files
    - Contain information entered into them by a user, an application or …
  - □ Directory Files
    - Manage the cataloging of the file system
  - □ Special Files (devices)
    - Used to access the peripheral devices
  - □ FIFO Files for Pipes

Extended File System



/

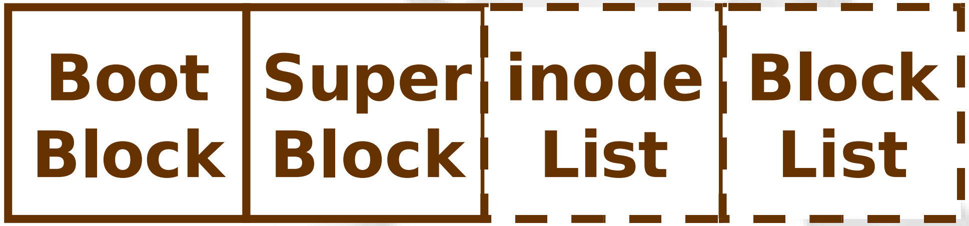
/bin /etc /dev ... /usr /home /root

ls ping

A B

data.txt pic.gif

File System Structure



Boot Block

Super Block

inode List

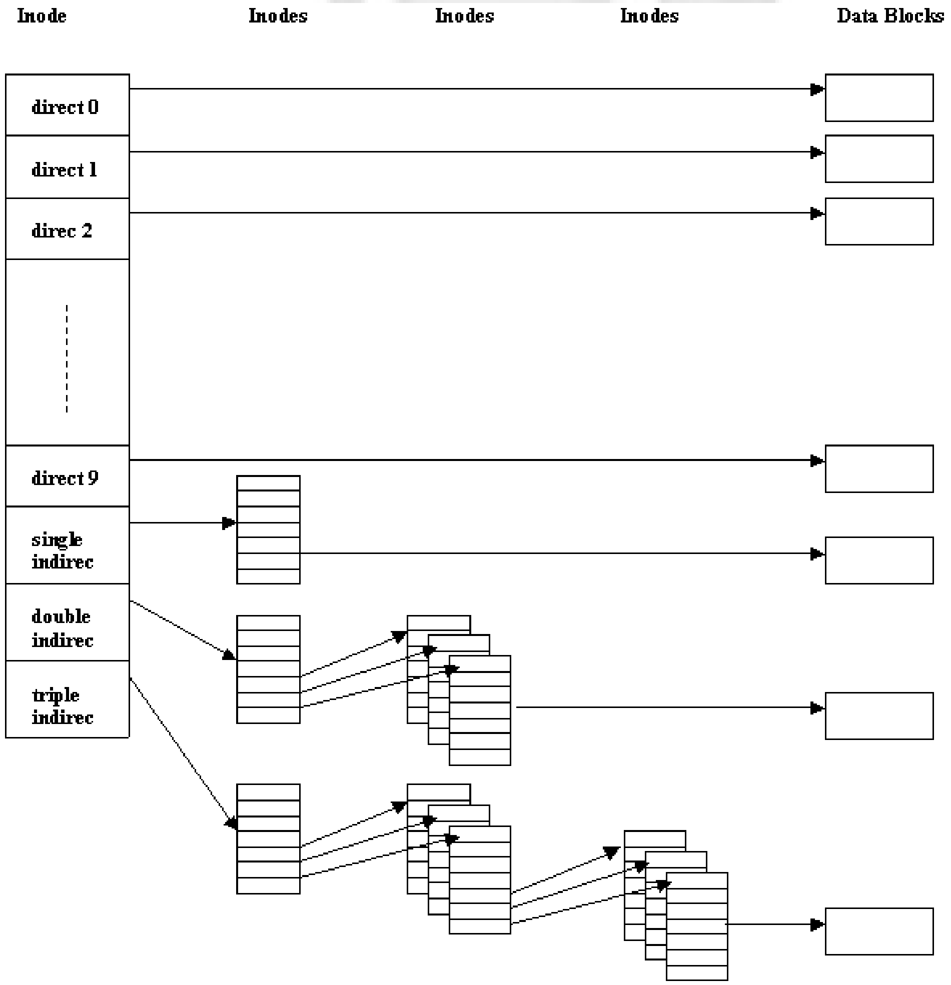
Block List

- Boot Block : information needs to boot the system
- Super Block : File System Specifications
  - □ Size
  - □ Max. number of files
  - □ Free blocks
  - □ Free inodes
- inode List
- Block List : The files data

Inode

- Each file has an inode structure that is identified by an i-number.
- The inode contains the information required to access the file.
- It doesn’t contain file name.

Inode (Cont.)


Inode
Inodes
Inodes
Inodes
Data Blocks
direct 0
direct 1
direc 2
direct 9
single indirec
double indirec
triple indirec

Directories

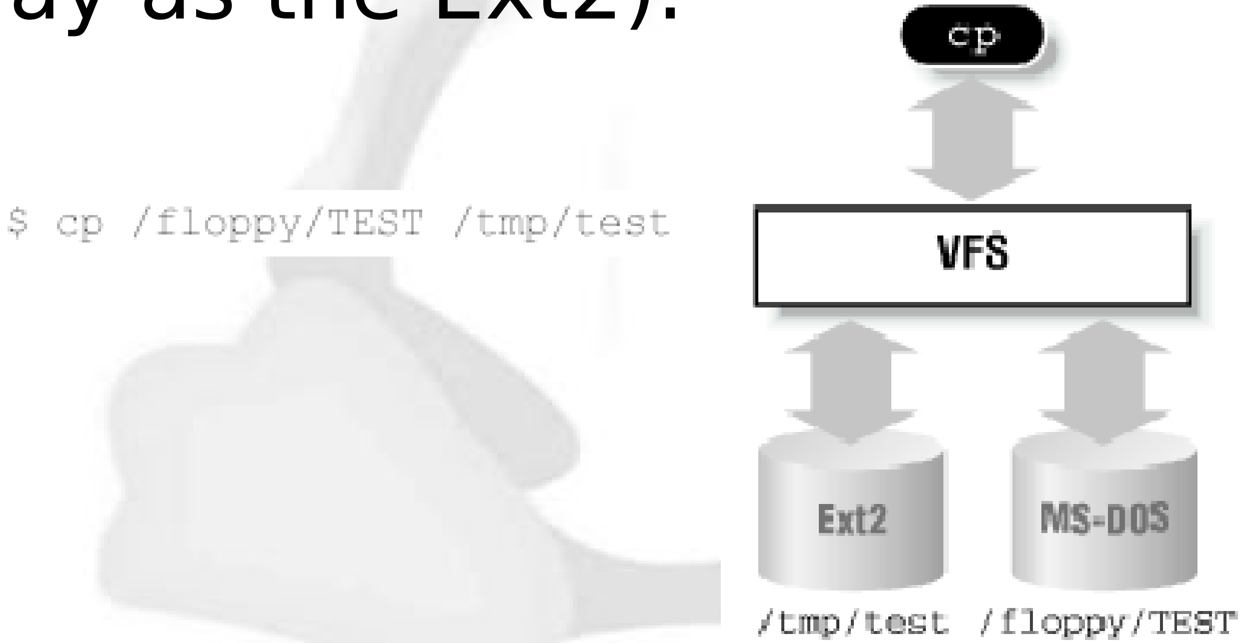
File Name	**inode Number**

Virtual File System

- It manages all the different file system.
- It is an abstraction layer between the application program and the file system implementations.

Virtual File System (Cont.)

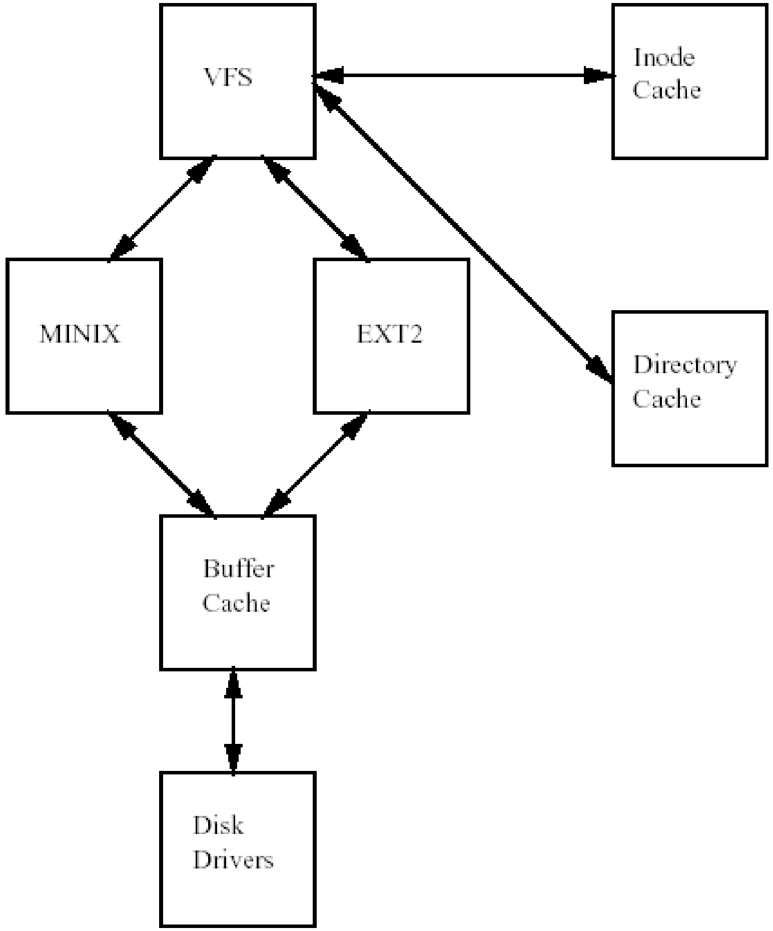
- It describes the system’s file in terms of superblocks and inodes (the same way as the Ext2).



```
$ cp /floppy/TEST /tmp/test
```

Virtual File System (Cont.)

- Inode cache
- Directory Cache



Contents

- What is Kernel ?
- Kernel Architecture Overview
  - □ User Space
  - □ Kernel Space
- Kernel Functional Overview
  - □ File System
  - □ Process Management
  - □ Device Driver
  - □ Memory Management
  - □ Networking

Process Management

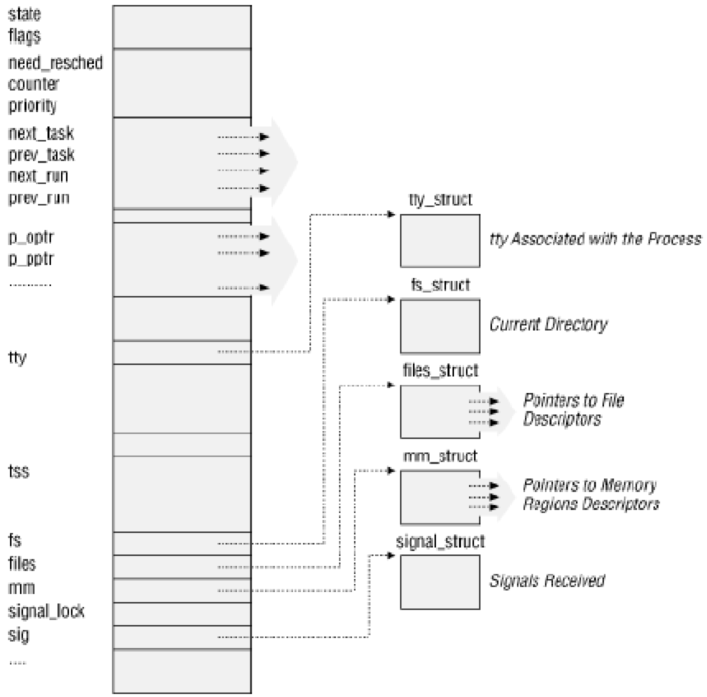
- The Unix OS is a time-sharing system.
- Every process is scheduled to run for a period of time (time slice).
- Kernel creates, manages and deletes the processes

Process Management (Cont.)

- Every process (except init) in the system is create as the result of a fork system call.
- The fork system call splits a process into two processes (Parent and Child).
- Each process has a unique identifier (Process ID).

Process Structure

- Each process is represented by a task_struct data structure.
  - □ It contains the specifications of each process such as:
    - State
    - Scheduling information
    - Identifier
    - …



Process Structure (cont.)

- The task_vector is an array of pointers to every task_struct data structure in the system.
  - This means that the maximum number of processes in the system is limited by the size of the task vector

Type of Processes

- Running
  - □ The process is either running or it is ready to run.
- Waiting
  - □ The process is waiting for an event or for a resource.
- Stopped
  - □ The process has been stopped, usually by receiving a signal.
- Zombie
  - □ This is a halted process which, for some reason, still has a task_struct data structure in the task vector.

Contents

- What is Kernel ?
- Kernel Architecture Overview
  - User Space
  - Kernel Space
- Kernel Functional Overview
  - File System
  - Process Management
  - Device Driver
  - Memory Management
  - Networking

Device Driver

- On of the purpose of an OS is to hide the system’s hardware from user.
- Instead of putting code to manage the HW controller into every application, the code is kept in the Linux kernel.
- It abstracts the handling of devices.
  - All HW devices look like regular files.

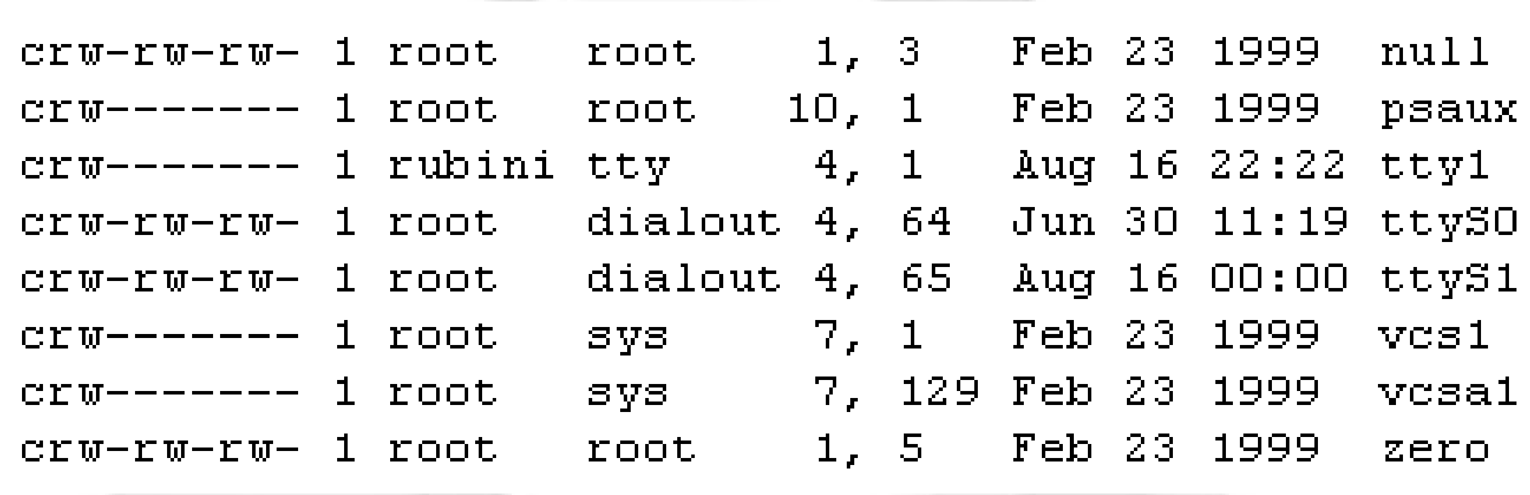
Type of devices

- Character devices
  - A character device is one that can be accessed as a stream of bytes.
  - Example : Keyboard, Mouse, …
- Block devices
  - A block device can be accessed only as multiples of a block.
  - Example : disk, …
- Network devices
  - They are created by Linux kernel.

Major Number and Minor Number

- Major Number
  - □ The major number identifies the driver associated with the device.
- Minor Number
  - □ The minor number is used only by the driver specified by the major number; other parts of the kernel don't use it.
  - □ It is common for a driver to control several devices, the minor number provides a way for the driver to differentiate among them.

Device Driver (Cont.)



```
crw-rw-rw- 1 root   root     1, 3   Feb 23 1999  null
crw------- 1 root   root    10, 1   Feb 23 1999  psaux
crw------- 1 rubini tty      4, 1   Aug 16 22:22 tty1
crw-rw-rw- 1 root   dialout  4, 64  Jun 30 11:19 ttyS0
crw-rw-rw- 1 root   dialout  4, 65  Aug 16 00:00 ttyS1
crw------- 1 root   sys      7, 1   Feb 23 1999  vcs1
crw------- 1 root   sys      7, 129 Feb 23 1999  vcsa1
crw-rw-rw- 1 root   root     1, 5   Feb 23 1999  zero
```

Contents

- What is Kernel ?
- Kernel Architecture Overview
  - User Space
  - Kernel Space
- Kernel Functional Overview
  - File System
  - Process Management
  - Device Driver
  - Memory Management
  - Networking

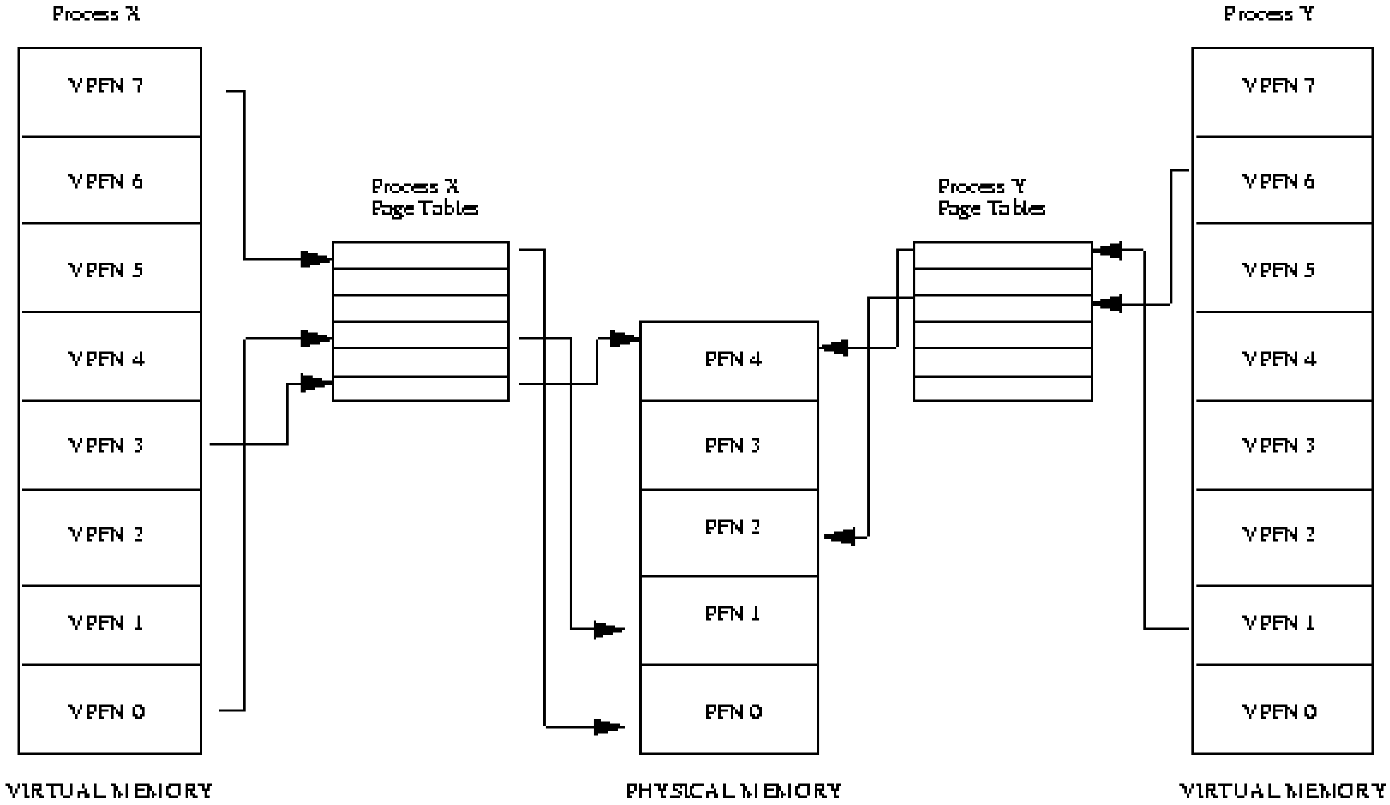
Memory Management

- Physical memory is limited.
- Virtual memory is developed to overcome this limitation.

Virtual memory

- Large Address space
- Protection
- Memory mapping
- Fair physical memory allocation
- Shared virtual memory

Physical and Virtual memory



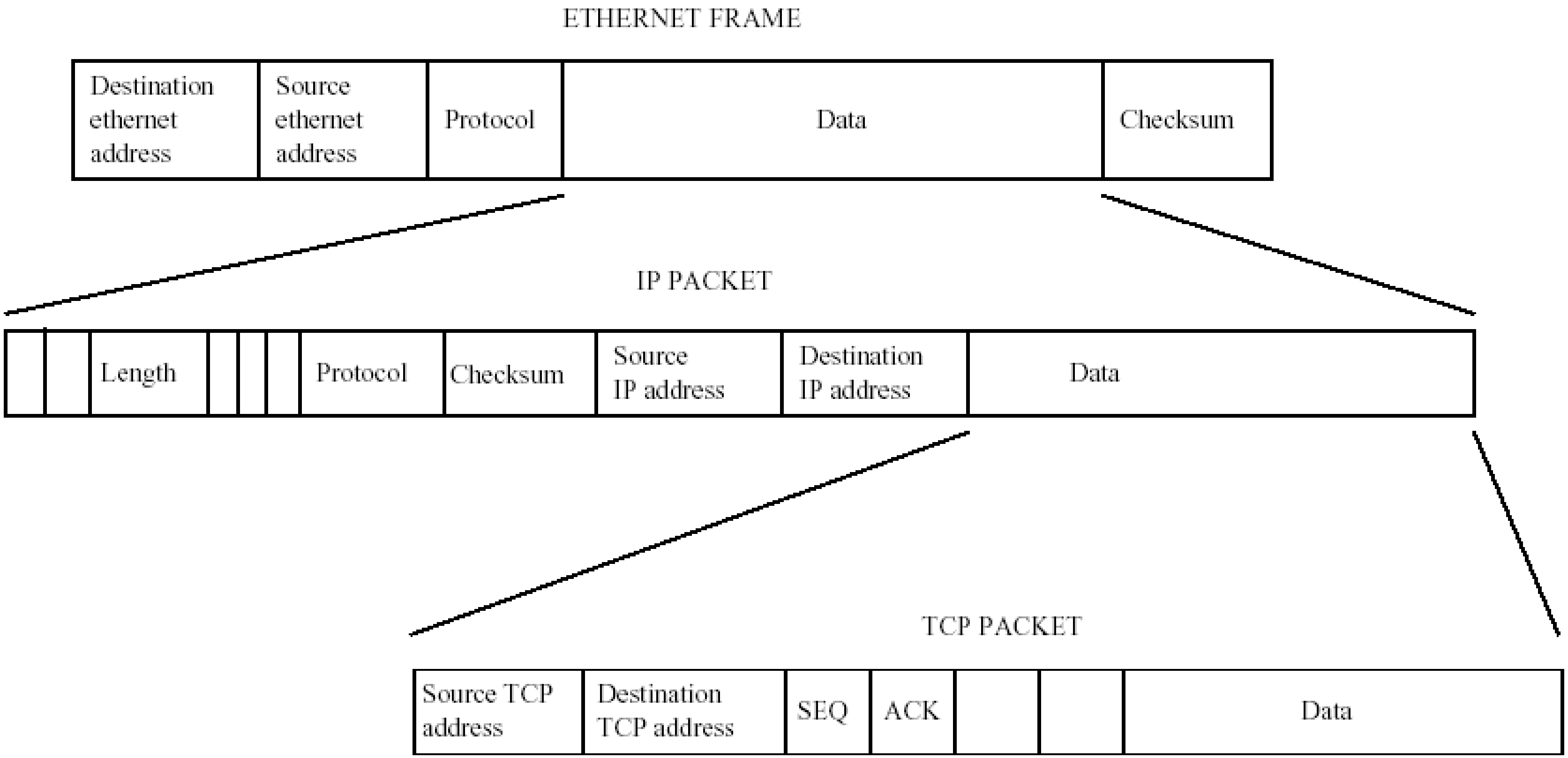
Swap memory

- It is a configurable partition on disk treated in a manner similar to memory.

Contents

- What is Kernel ?
- Kernel Architecture Overview
  - □ User Space
  - □ Kernel Space
- Kernel Functional Overview
  - □ File System
  - □ Process Management
  - □ Device Driver
  - □ Memory Management
  - □ Networking

Network layers



ETHERNET FRAME

Destination ethernet address | Source ethernet address | Protocol | Data | Checksum

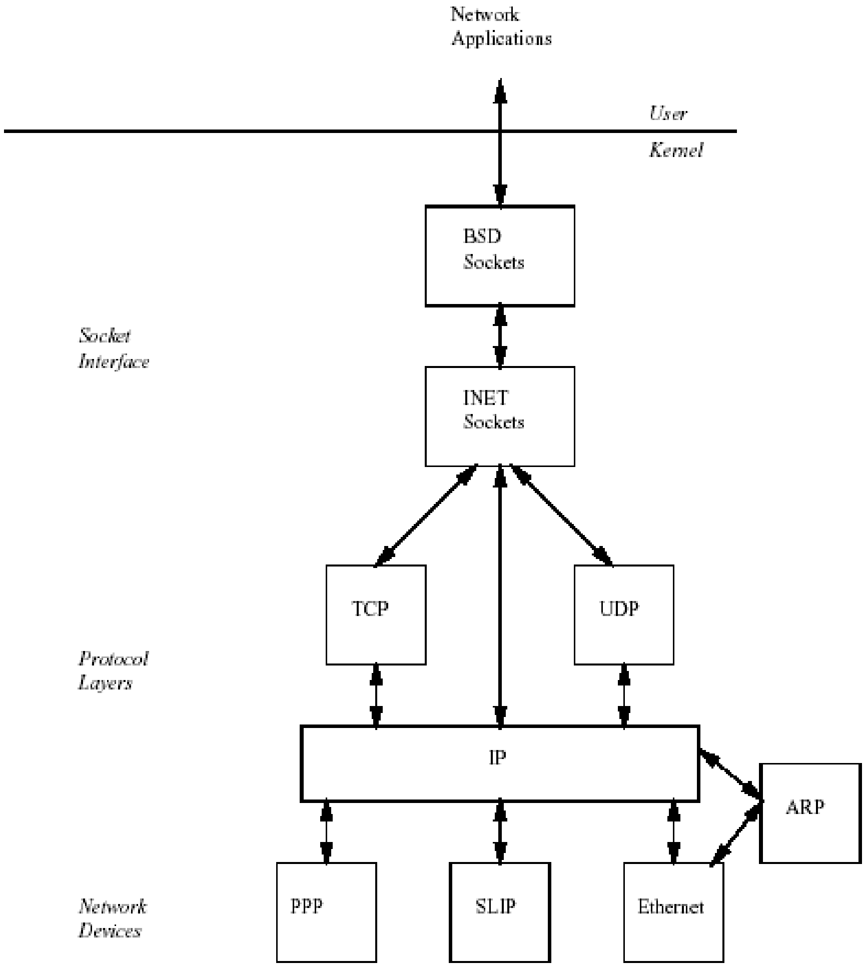
IP PACKET

Length | Protocol | Checksum | Source IP address | Destination IP address | Data

TCP PACKET

Source TCP address | Destination TCP address | SEQ | ACK | Data

Linux network layers



BSD socket layer

- It is a general interface (abstract layer).
    - □ Used in networking and IPC.
- Socket address families:
    - □ UNIX
    - □ INET
    - □ AX25
    - □ IPX
    - □ APPLETALK
    - □ X25

What is socket?

```
main()
{
    FILE *fd;
    fd = fopen (…);
    process (fd);
    fclose (fd);
}
```

```
main()
{
    int sockfd;
    sockfd = socket (…);
    process (sockfd);
    close (sockfd);
}
```

INET socket layer

- It supports the Internet address family.
- Its interface with BSD socket layer is through a set of operation which is registered with BSD socket layer.

Type of sockets

- Stream Socket
  - Provide a reliable, sequenced, two-way connection (such as TCP).
- Datagram Socket
  - A connection-less and unreliable connection (such as UDP).
- Raw Socket
  - Used for internal network protocols.

Question?