JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

Lecture-33 By Vivek Dubey

COMPUTER SCIENCE AND ENGINEERING

CS403PC

COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

CS403PC

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Scheme of Operating System

S. No.	Course Code	Course Title	L	T	P	Credits
3	CS403PC	Operating Systems	3	0	0	3
6	CS406PC	Operating Systems Lab	0	0	3	1.5

Lecture-33 By Vivek Dubey

UNIT-5: File System Interface and Operations

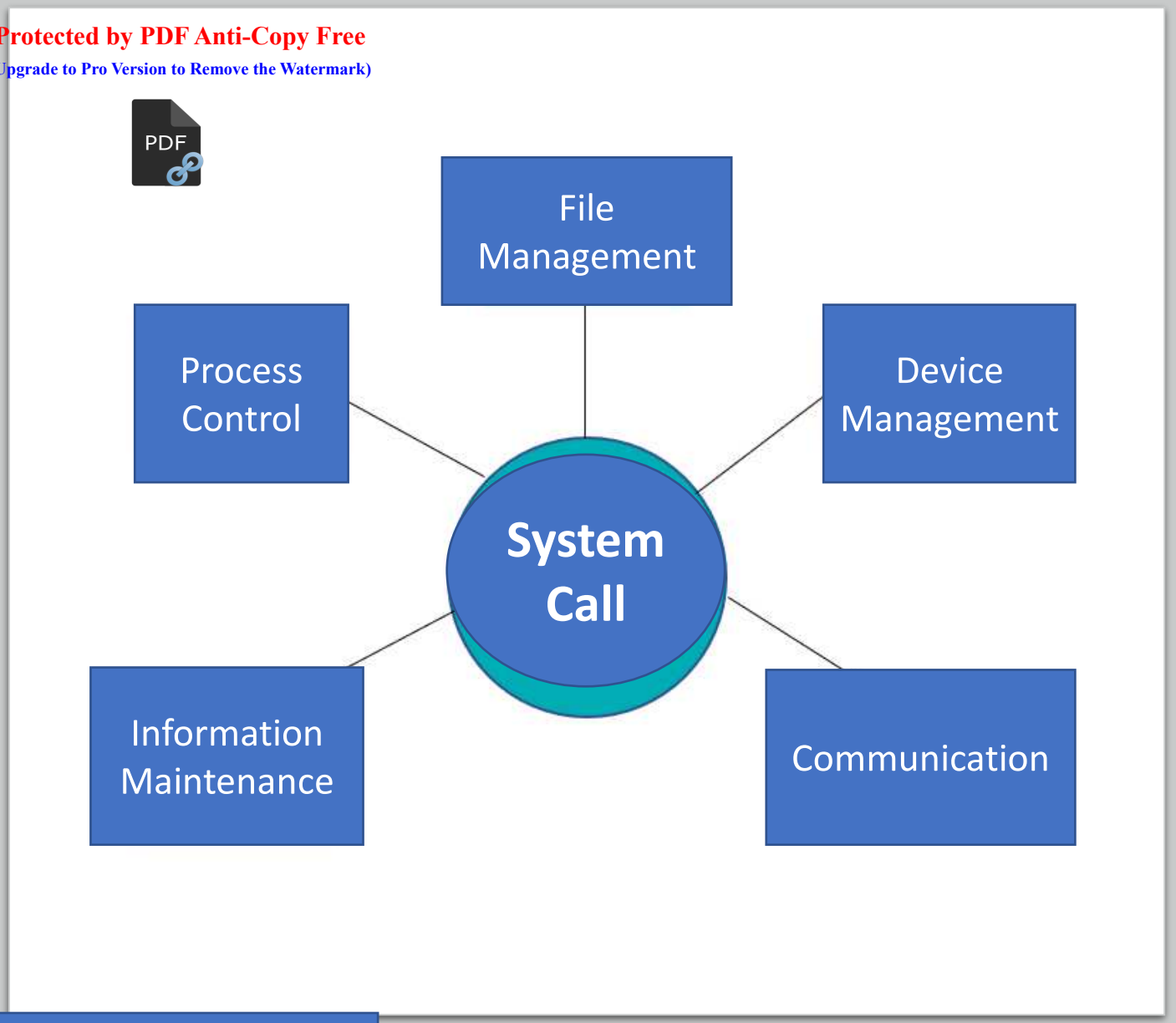
- Access methods,
- Directory Structure,
- Protection,
- File System Structure,
  - Allocation methods,
  - Free-space Management.
- Usage of open, create, read, write, close, lseek, stat, ioctl system calls.

Lecture-33 By Vivek Dubey

System Calls in OS

System calls are divided into 5 categories mainly :

- Process Control
- File Management
- Device Management
- Information Maintenance
- Communication



Process Control

Lecture-33 By Vivek Dubey

- **fork()**
  - **A new process is created by** *the fork() system call*.
  - **A new process may be created with fork() without a new program** being run-the new sub-process simply continues to execute exactly the same program that the first (parent) process was running.
  - **It is one of the most widely used system calls** *under process management.*
- **exit()**
  - **The exit() system call is used by** *a program to terminate its execution.*
  - **The operating system reclaims resources that were used by** *the process after the exit() system call.*
- **exec()**
  - **A new program will start executing after** *a call to exec()*
  - **Running a new program does not require that a new process be created first**:
    - any process may call exec() at any time. The currently running program is immediately terminated, and the new program starts executing in the context of the existing process.

File Management

Lecture-33 By Vivek Dubey

- **File Management System Calls** *handle file manipulation jobs* like creating a file, reading, and writing, etc.
- The Linux System calls under this are
  - **open(),**
  - **read(),**
  - **write(),**
  - **close().**

Contd..

- **open():**
  - **It is the system call** *to open a file*.
  - **This system call just** *opens the file*, to perform operations *such as* read and write.
- **close():**
  - **This system call** *closes the opened file*.

Contd..

- **read():**
  - **This system call** *opens the file in reading mode.*
  - **We can not edit the files** *with this system call.*
  - **Multiple processes can execute** *the read() system call* on the same file simultaneously.
- **write():**
  - **This system call** *opens the file in writing mode.*
  - **We can edit the files** *with this system call.*
  - **Multiple processes can not execute** *the write() system call* on the same file simultaneously.

Device Management

- **Device Management** *does the job of device manipulation* like reading from device buffers, writing into device buffers, etc. The Linux System calls under this is **ioctl().**
- **ioctl():**
  - ioctl() is referred to as Input and Output Control.
  - ioctl is a system call for device-specific input/output operations and other operations which cannot be expressed by regular system calls.

Information Maintenance

Lecture-33 By Vivek Dubey

- **It handles information** and **its transfer** *between the OS and the user program*. In addition, **OS keeps the information** *about all its processes* and **system calls are used** *to access this information.* The System calls under this are **getpid(), alarm(), sleep().**
- **getpid():**
  - **getpid stands for** *Get the Process ID*.
  - **The getpid() function** *shall return the process ID* of the calling process.
  - **The getpid() function** shall always be successful and **no return value** *is reserved to indicate an error*.

Contd..

- **alarm():**
  - **This system call sets an alarm clock** *for the delivery of a signal* that when it has to be reached.
  - **It arranges for a signal to be delivered** *to the calling process*.
- **sleep():**
  - **This System call** *suspends the execution* of the currently running process f*or some interval of time*
  - **Meanwhile**, *during this interval*, another process is given chance to execute

Communication

- **These types of system calls** *are specially* used for inter-process communications.
- **Two models are used** *for inter-process communication*
  1. Message Passing(processes exchange messages with one another)
  2. Shared memory(processes share memory region to communicate)
- The system calls under this are **pipe() , shmget() ,mmap().**

Contd..

- **pipe():**
  - **The pipe() system call is used** *to communicate between different Linux processes.*
  - **It is mainly used** *for inter-process communication.*
  - **The pipe() system function** *is used to open file descriptors.*
- **shmget():**
  - **shmget stands for** *shared memory segment.*
  - **It is mainly used** *for Shared memory communication.*
  - **This system call is used** *to access the shared memory* and **access the messages** *in order to communicate with the process.*
- **mmap():**
  - **This function call is used** *to map or unmap files or devices into memory.*
  - **The mmap() system call is responsible** *for mapping the content of the file to the virtual memory space of the process.*