Session 22

Process, PCB (Process Control Block), and File Descriptor, which are fundamental in understanding how Linux (or any OS) manages programs and resources.

1. Process

Definition:

- A process is an instance of a program in execution.
- Every running program in Linux becomes a process, which is an active entity with:
- Its own memory
- CPU registers
- File descriptors
- Program counter
- Process ID (PID)

Types:

Foreground Process: Takes input from terminal.

Background Process: Runs without terminal interaction (& in shell).

Lifecycle of a Process:

- New – Process is being created
- Ready – Waiting to be assigned to a processor
- Running – Instructions are being executed
- Waiting – Waiting for some event (e.g., I/O)
- Terminated – Finished execution

2. PCB (Process Control Block)

Definition:

The PCB is a data structure used by the OS to store all information about a process.

Contains:

Component	**Description**
PID	Unique process ID
Program Counter	Address of next instruction to execute

CPU Registers	Values of all CPU registers
Process State	Ready, Running, Waiting, etc.
Memory Info	Base and limit of allocated memory
Open File Descriptors	Table of files the process has opened
Scheduling Info	Priority, timeslice, etc.
Accounting Info	CPU used, time limits, etc.
Parent/Child Info	Process tree information

Analogy:

Think of PCB as the "resume" or "profile" of a process. When the OS switches between processes (context switching), it saves and restores this data.

3. File Descriptor (FD)

Definition:

A file descriptor is a non-negative integer that uniquely identifies an open file within a process.

In Linux:

Every process has a file descriptor table

File descriptors point to entries in a system-wide open file table

Standard File Descriptors:

FD	Description
0	Standard Input (stdin)
1	Standard Output (stdout)
2	Standard Error (stderr)

When are FDs used?

When you open a file using system calls like open(), pipe(), or socket(), the system returns an FD.

int fd = open("data.txt", O_RDONLY);

Example:

write(1, "Hello\n", 6); // Writes to stdout

You can also use:

dup() or dup2() to duplicate file descriptors

fcntl() to manipulate them

close(fd) to release them

How They Relate:

Concept	**Relation to Others**
Process	A running program with resources and identity
PCB	Stores all info the OS needs about a process
File Descriptor	Tracked inside PCB; represents open files/devices

Summary Table

TermMeaning		**OS Role**
Process	Executing program	Fundamental execution unit
PCB	Data structure for process metadata	Used for context switching, scheduling
File Descriptor	Handle for open file in process	Used for I/O operations (read/write/close)