**Chapter 3: Operating systems concept**

1. **Process concept**

* A process is a program in execution, and the status of the current activity of

a process is represented by the program counter, as well as other registers.

• The layout of a process in memory is represented by four different sections:

(1) text, (2) data, (3) heap, and (4) stack.

• As a process executes, it changes state. There are four general states of a

process: (1) ready, (2) running, (3) waiting, and (4) terminated.

• A process control block (PCB) is the kernel data structure that represents a

process in an operating system.

1. **Process scheduling**

• The role of the process scheduler is to select an available process to run on

a CPU.

1. **Operation on process**

• An operating system performs a context switch when it switches from

running one process to running another.

1. **Interprocess communication**

• The fork() and CreateProcess() system calls are used to create processes on UNIX and Windows systems, respectively.

1. **IPC in shared-memory system**

• When shared memory is used for communication between processes, two

(or more) processes share the same region of memory. POSIX provides an

API for shared memory.

• Two processes may communicate by exchanging messages with one

another using message passing. The Mach operating system uses message

passing as its primary form of interprocess communication. Windows

provides a form of message passing as well.

1. **IPC in message-passing systems**

• A pipe provides a conduit for two processes to communicate. There are

two forms of pipes, ordinary and named. Ordinary pipes are designed for

communication between processes that have a parent–child relationship.

Named pipes are more general and allow several processes to communicate.

**7 .Example of IPC systems**

• UNIX systems provide ordinary pipes through the pipe() system call.

Ordinary pipes have a read end and a write end. A parent process can, for

example, send data to the pipe using its write end, and the child process

can read it from its read end. Named pipes in UNIX are termed FIFOs.

• Windows systems also provide two forms of pipes—anonymous and

named pipes. Anonymous pipes are similar to UNIX ordinary pipes. They

are unidirectional and employ parent–child relationships between the

communicating processes. Named pipes offer a richer form of interprocess

communication than the UNIX counterpart, FIFOs.

**8. Communication in client-server**

• Two common forms of client–server communication are sockets and

remote procedure calls (RPCs). Sockets allow two processes on different

machines to communicate over a network. RPCs abstract the concept of

function (procedure) calls in such a way that a function can be invoked on

another process that may reside on a separate computer.

• The Android operating system uses RPCs as a form of interprocess communication using its binder framework