Chapter3.
Process Concept

Contents

- Process Concept
- Process Scheduling
- Operations On Process
- Inter-Process Communication
- Communication in Client-Server Systems

Process Concept

Process

Process VS Program

Process in Memory

Process State

Process Control Block

Threads

Process

- A program in execution.
- Other word : job, user program, task
- It is more than the program code.
  It is not a program!

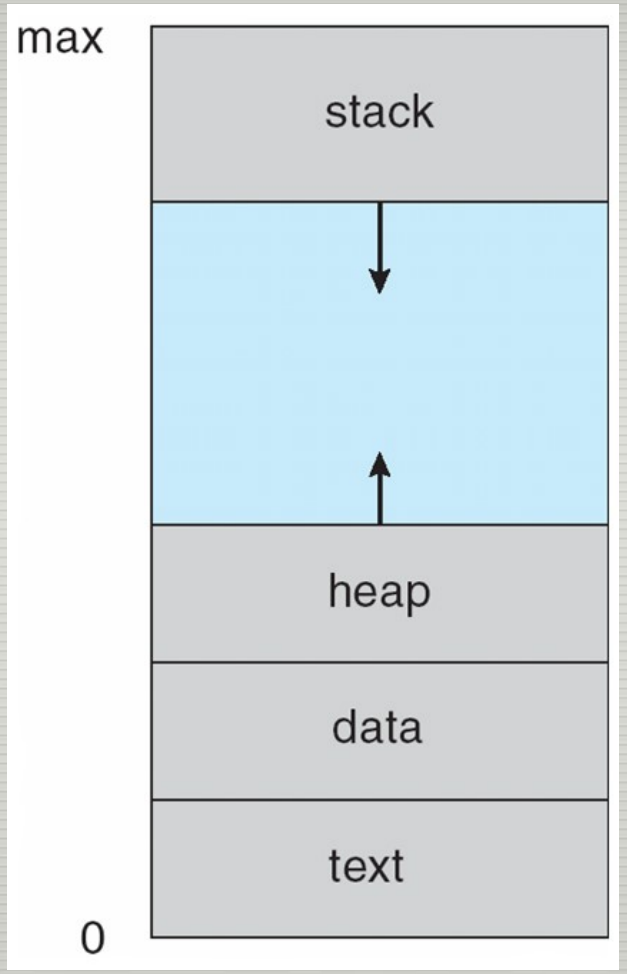
Process VS Program

- Active Entity
- A program becomes when program loaded into memory.
- Includes program counter, stack, data section, heap...
- Can be more than one entities.

- Passive Entity
- Executable file stored in disk.
- Has a text section only.
- Can not be only one entity.

Process	Program

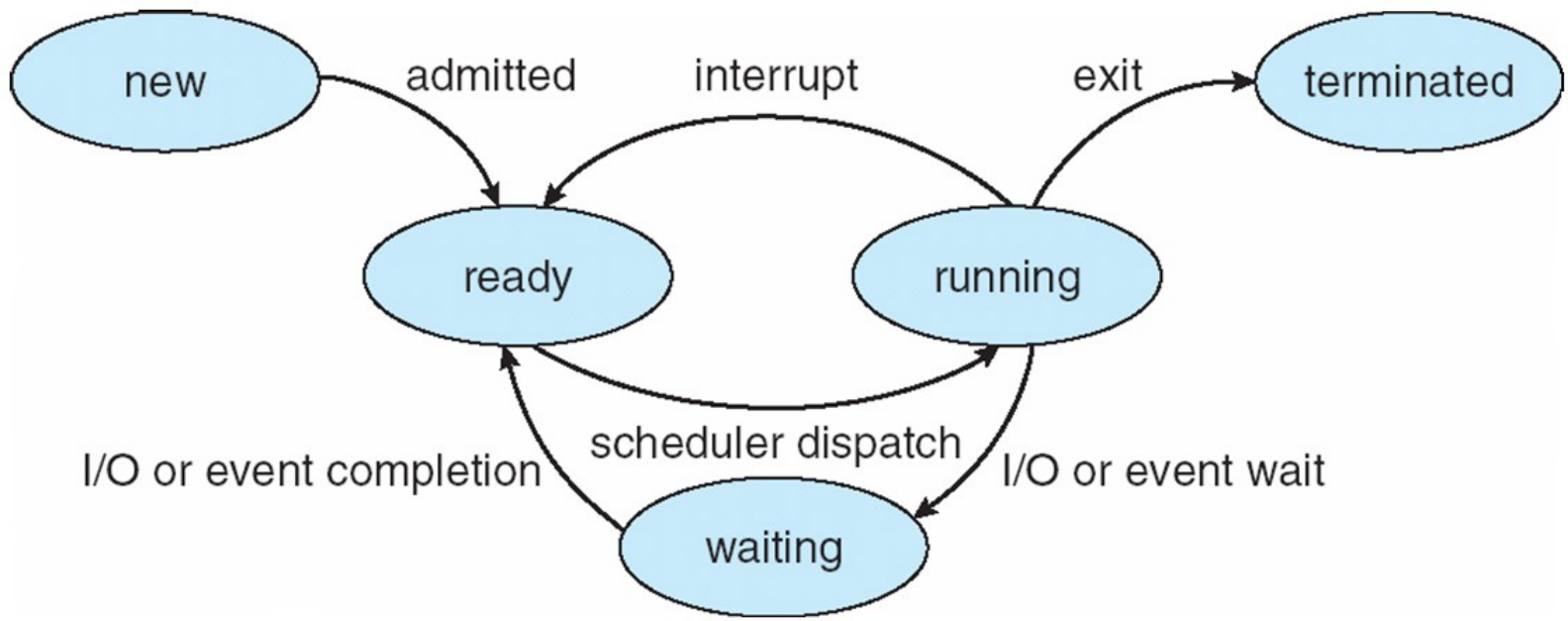
Processes in Memory



Process State

- As a process executes, it changes state.
- **New** : The processes is being created.
- **Running** : Instructions are being executed.
- **Waiting** : The process is waiting for some event to occur.
- **Ready** : The process is waiting to be assigned to a processor.
- **Terminated** : The process has finished execution.

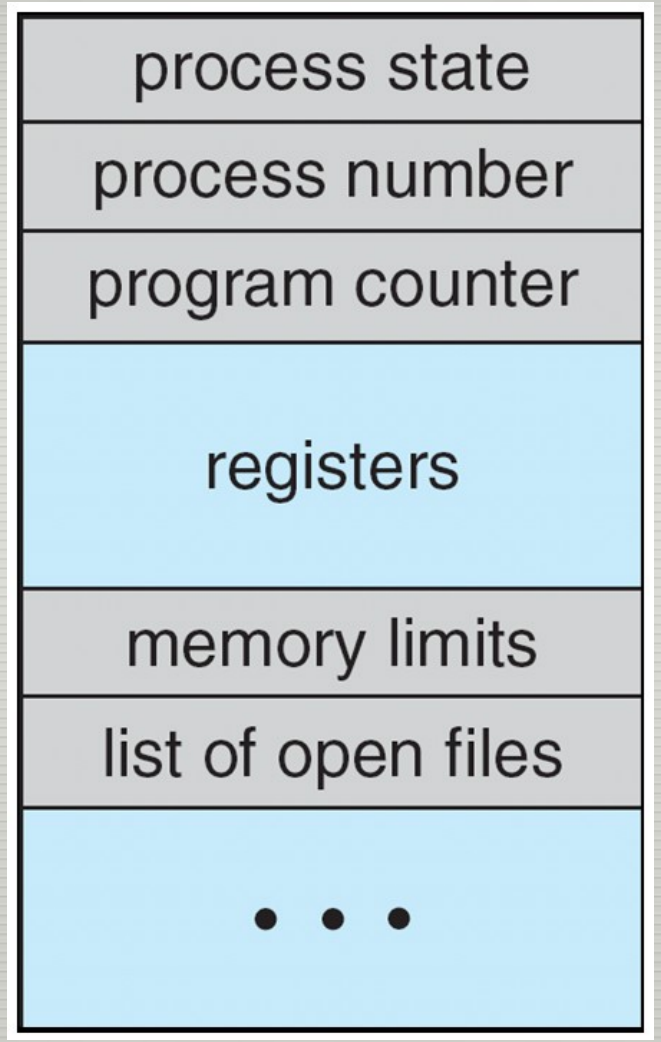
Diagram of Process State



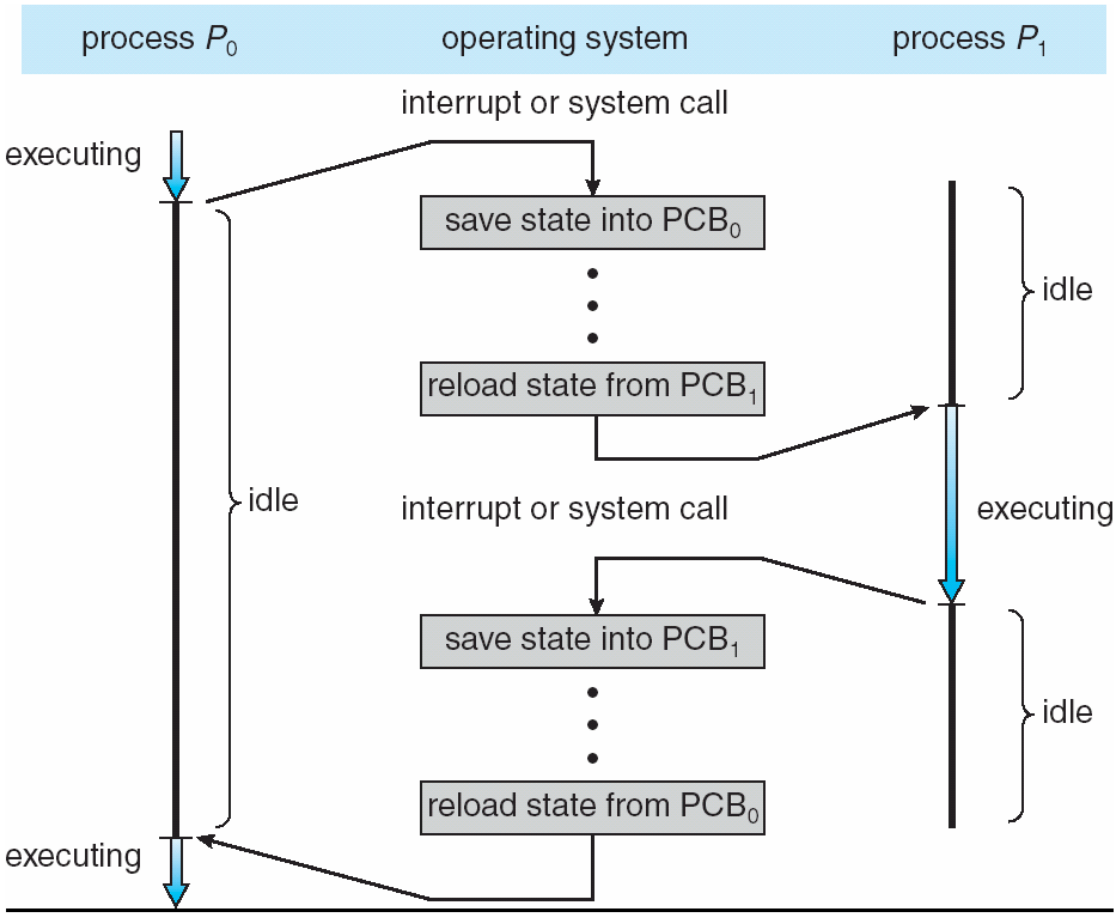
Process Control Block(PCB)

- The repository for any information that may vary from process to process.
- **Process state**
- **Program counter**
- **CPU registers**
- **CPU-scheduling information**
- **Memory-management information**
- **Accounting information**
- **I/O status information**

Process Control Block(PCB)



CPU Switch From Process to Process



Threads

- A thread is a flow of control in a execution.
- A single thread of control allows the process to perform only one tasks at one time.
- Multi-threads perform more than one task at time.
- On a system that supports threads, the PCB is extended to include information for each thread.(Other information are also changed to support threads)

Process Scheduling

Scheduling & Process Scheduler

Scheduling Queues

Schedulers

Context Switching

Scheduling & Process Scheduler

- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
- To maximize CPU utilization, a proper process should be selected to be performed by CPU. – Scheduling
- Process Scheduler selects an available process for program execution on the CPU.

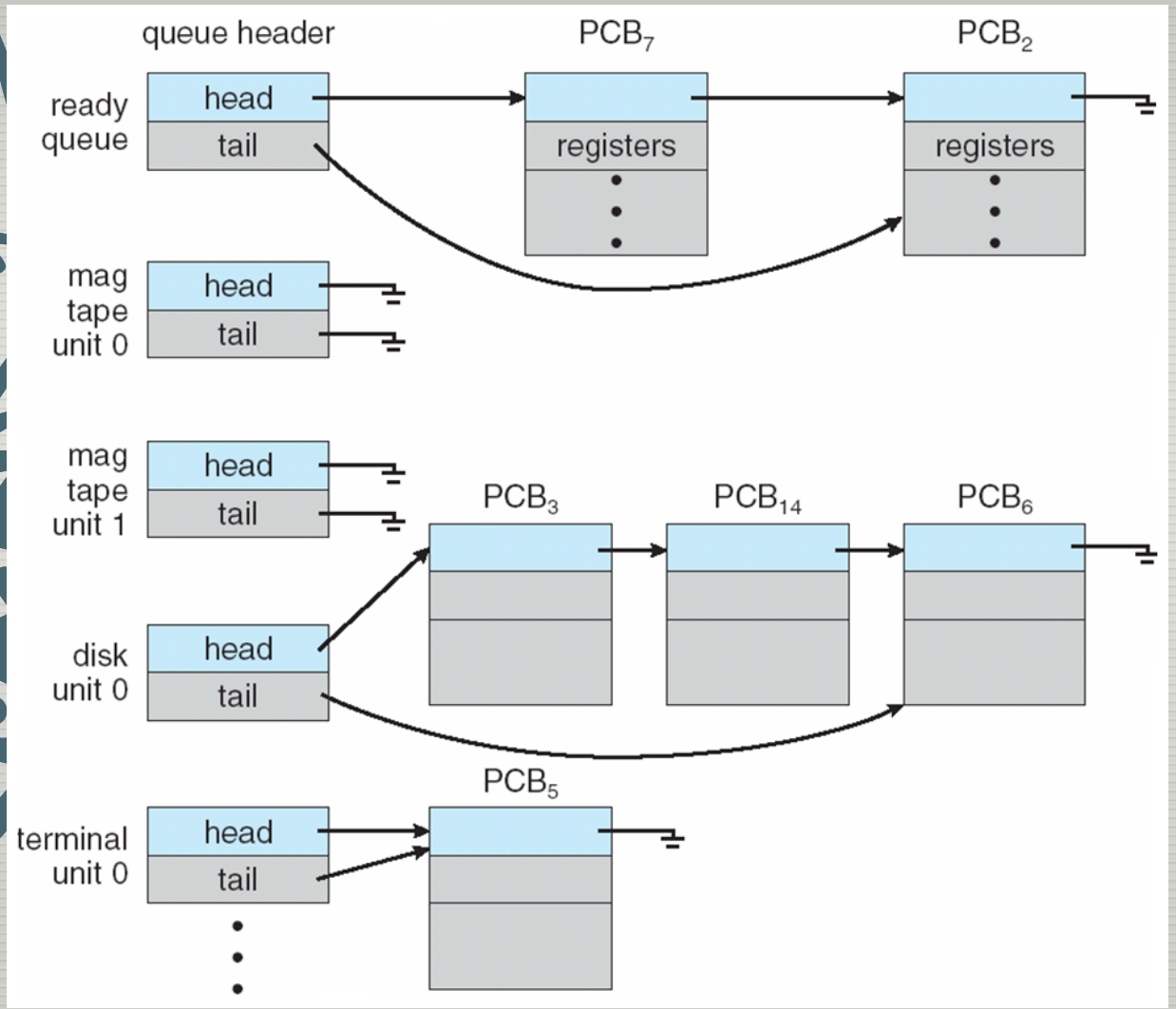
Scheduling Queues

- **Job Queue** : Set of all processes in the system.
- **Ready Queue** : Set of all processes residing in main memory ready and waiting to execute.
- **Device Queue** : Set of processes waiting for an I/O device.
- And so on.
- Processes migrate among the various queues.

Scheduling Queues

- As processes enter the system, they are put into a job queue, which consist of all processes in the system.
- Queues are generally stored as a linked list of PCBs.
- Queueing diagram : A common representation of process scheduling.

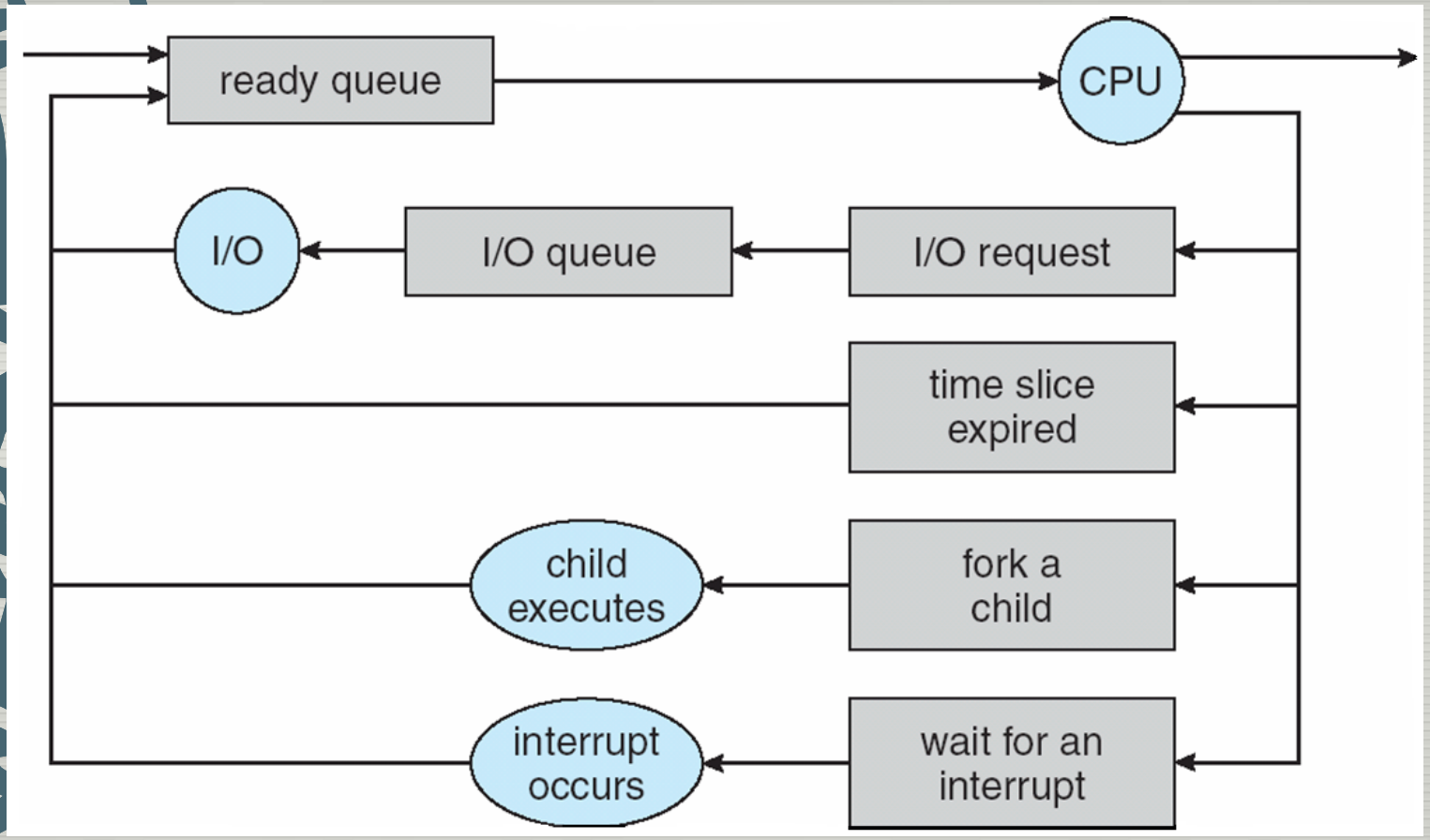
The Ready Queue and various I/O Device Queues



Queueing Diragram

- Rectangular box : A queue.
- Circle : The resources that serve the queues.
- Arrow : flow of processes in the system.

Queueing Diagram representation of process scheduling



Schedulers

- The operating system must select processes from queues for scheduling purposes.
- The scheduler selects the process to execute.
- **Long term scheduler(job** scheduler) : selects which processes should be brought into the ready queue.
- **Short term scheduler(CPU** scheduler) : selects which process should be executed next and allocates CPU.

Long Term Scheduler

- Selects which processes should be brought into the ready queue(memory) from pool.
- Controls the degree of multiprogramming.
- **Degree of multiprogramming** : The number of processes in memory.
- Executes much less frequently

Short Term Scheduler

- Selects a process from ready queue and allocates the CPU.
- It must select a new processes for the CPU frequently.
- It should be fast.

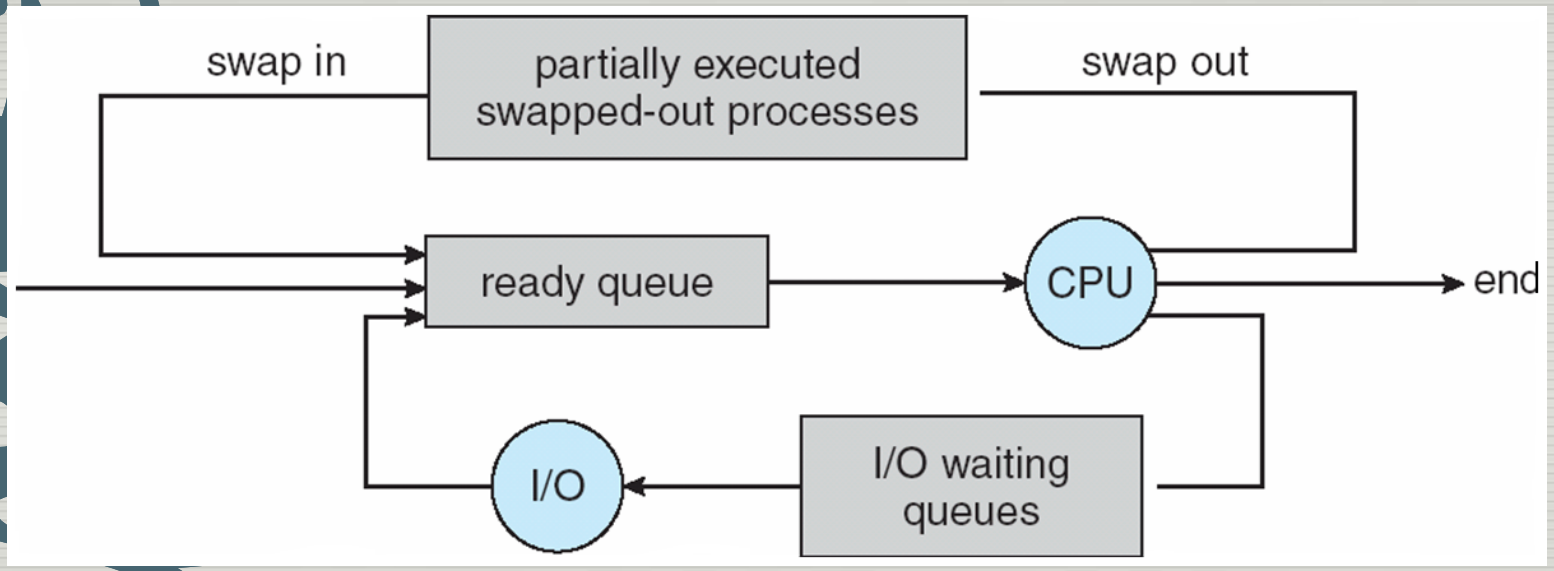
I/O bound process & CPU bound process

- **I/O bound process** : A process that spends more of its time doing I/O than it doing computations.
- **CPU Bound process** : Generates I/O requests infrequently, using more of its time doing computations.
- Long term scheduler should select a good **process mix** of I/O bound and CPU bound processes.

Mid-Term Scheduler

- Key Idea : It can be advantageous to reduce the degree of multiprogramming.
- **Swapping** is used to improve the process mix or because a change in memory requirements has overcommitted available memory.

Mid-Term Scheduler



Context Switching

- System switches the running process to another one because of some reasons.(ex : interrupt)
- When switching occurs, the system should save the context of the running process – Context Switching.
- The context is represented in the PCB of process.
- **Context switching time is overhead** : The system does no useful work while switching.

Operations on Processes

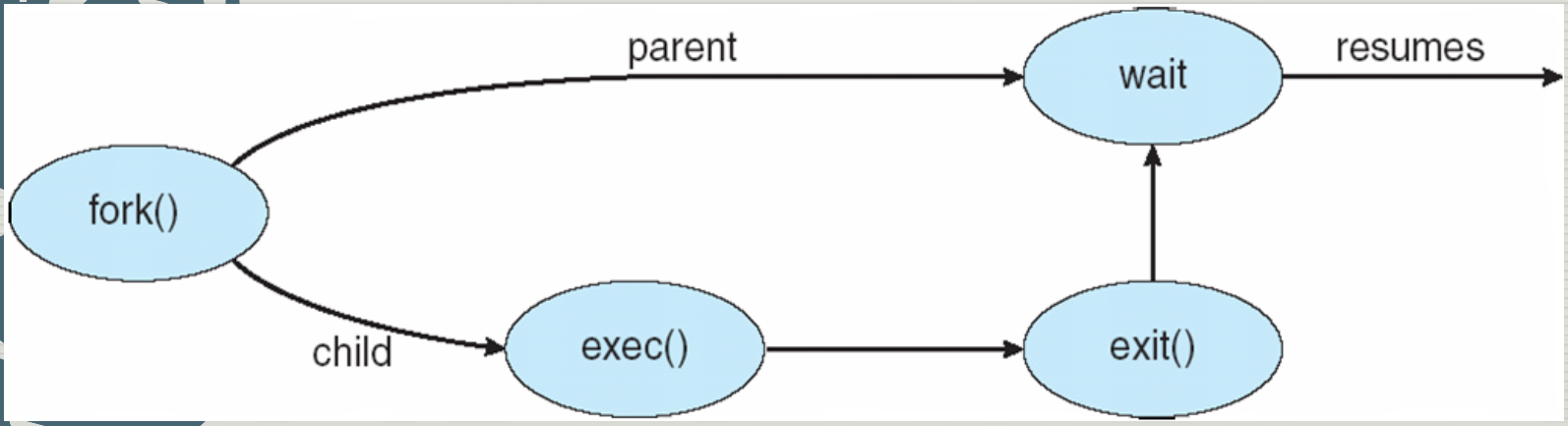
Process Creation

Process Termination

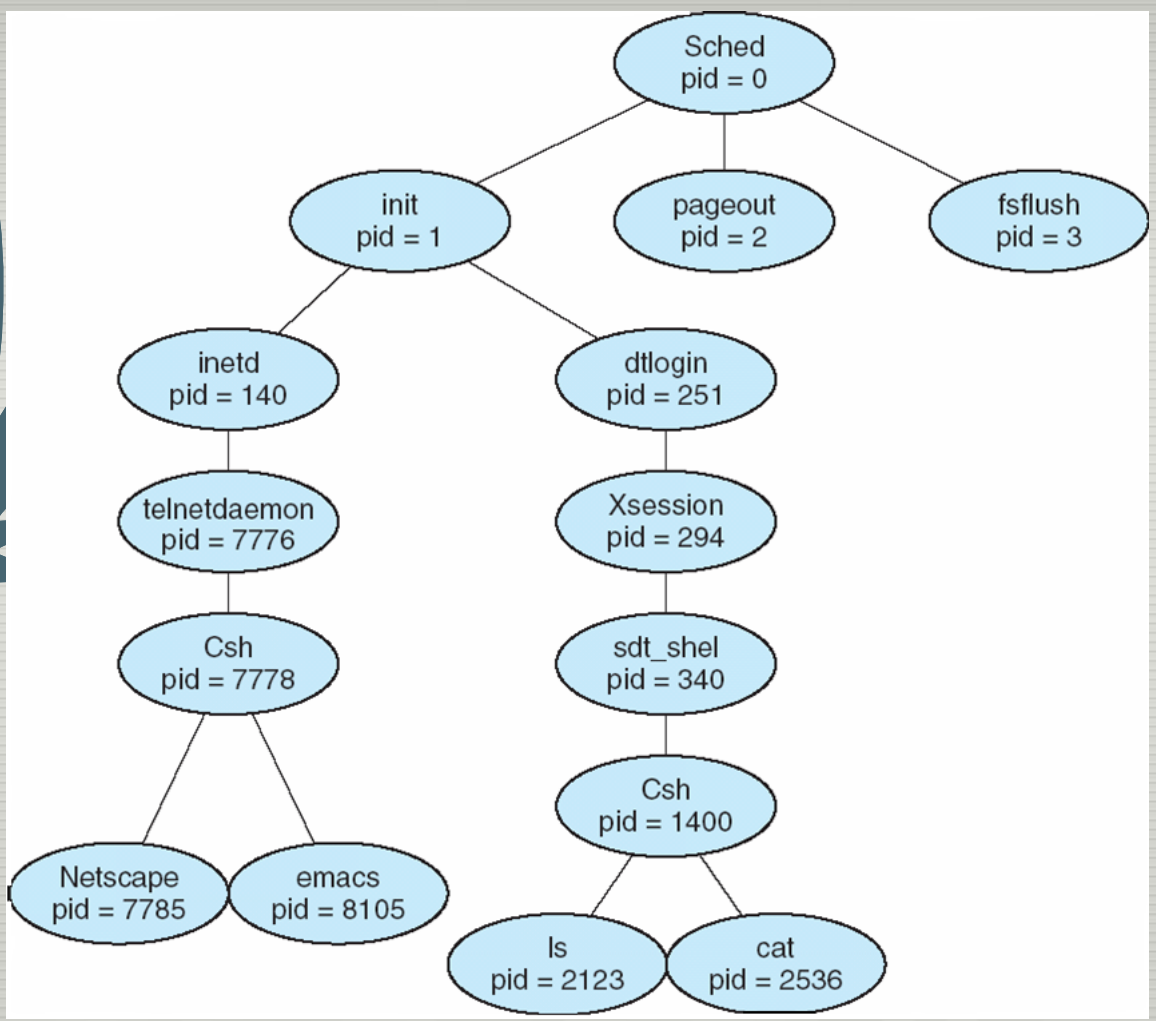
Process Creation

- A Parent process can create child processes.
- Most operating systems identify processes according to a unique process identifier(pid).
- **Resource Sharing** : Share all / subset of / no parent's resources.
- **Execution after spawning** : Execute concurrently. / Parent waits for termination of child process.
- **Address space** : Child duplicates of parent's./ Load a new program into it.

Example : UNIX



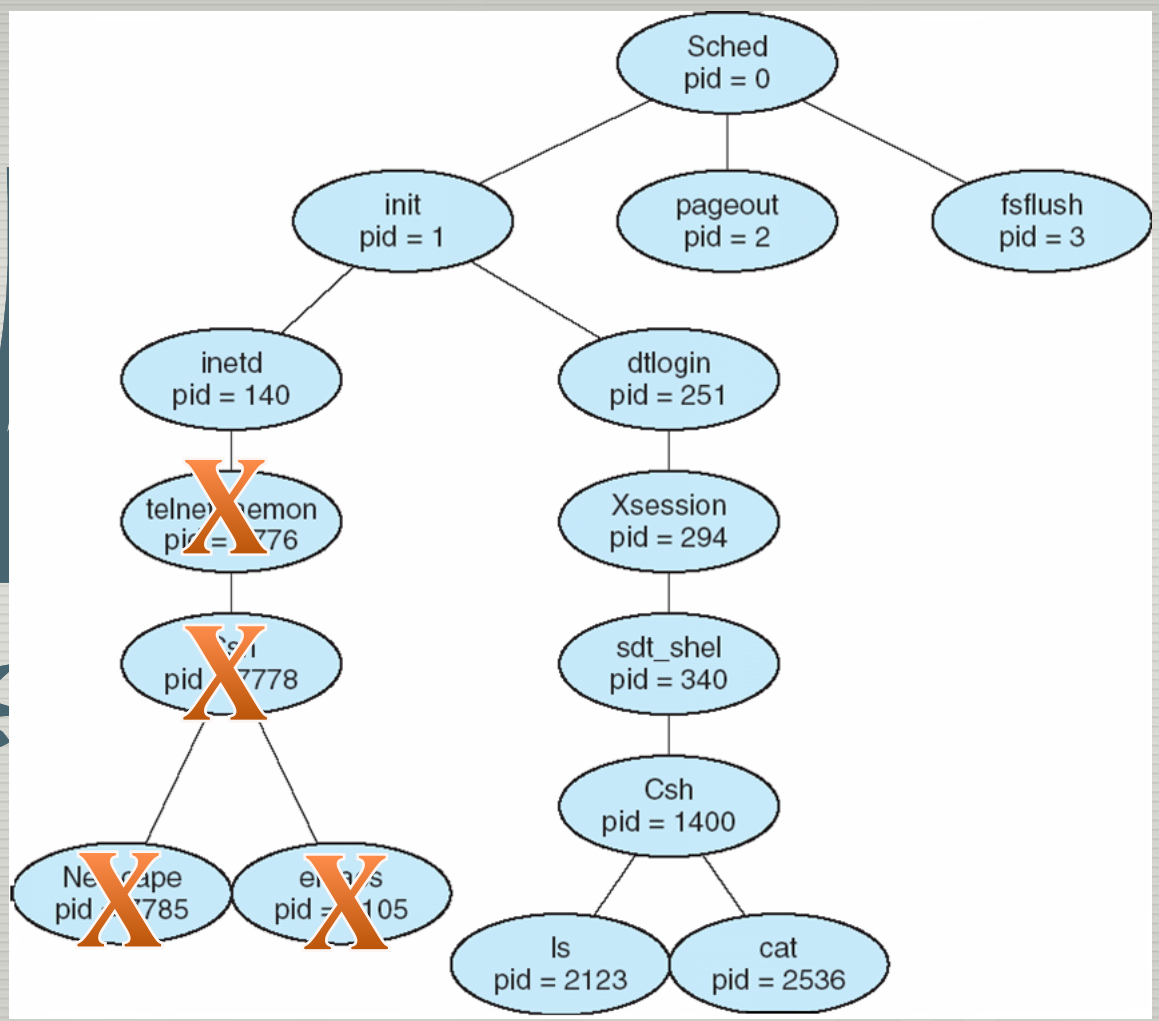
Example : UNIX



Process Termination

- Executing its final statements.
- Asking the operating system to delete a process.(Ex : exit() system call)
- Parent asks for operating system to terminate its child process.

Cascading Terminatiaon



Inter-process Communication

Inter-Process Communication

Shared Memory Systems

Producer – Consumer Problem

Message-Passing Systems

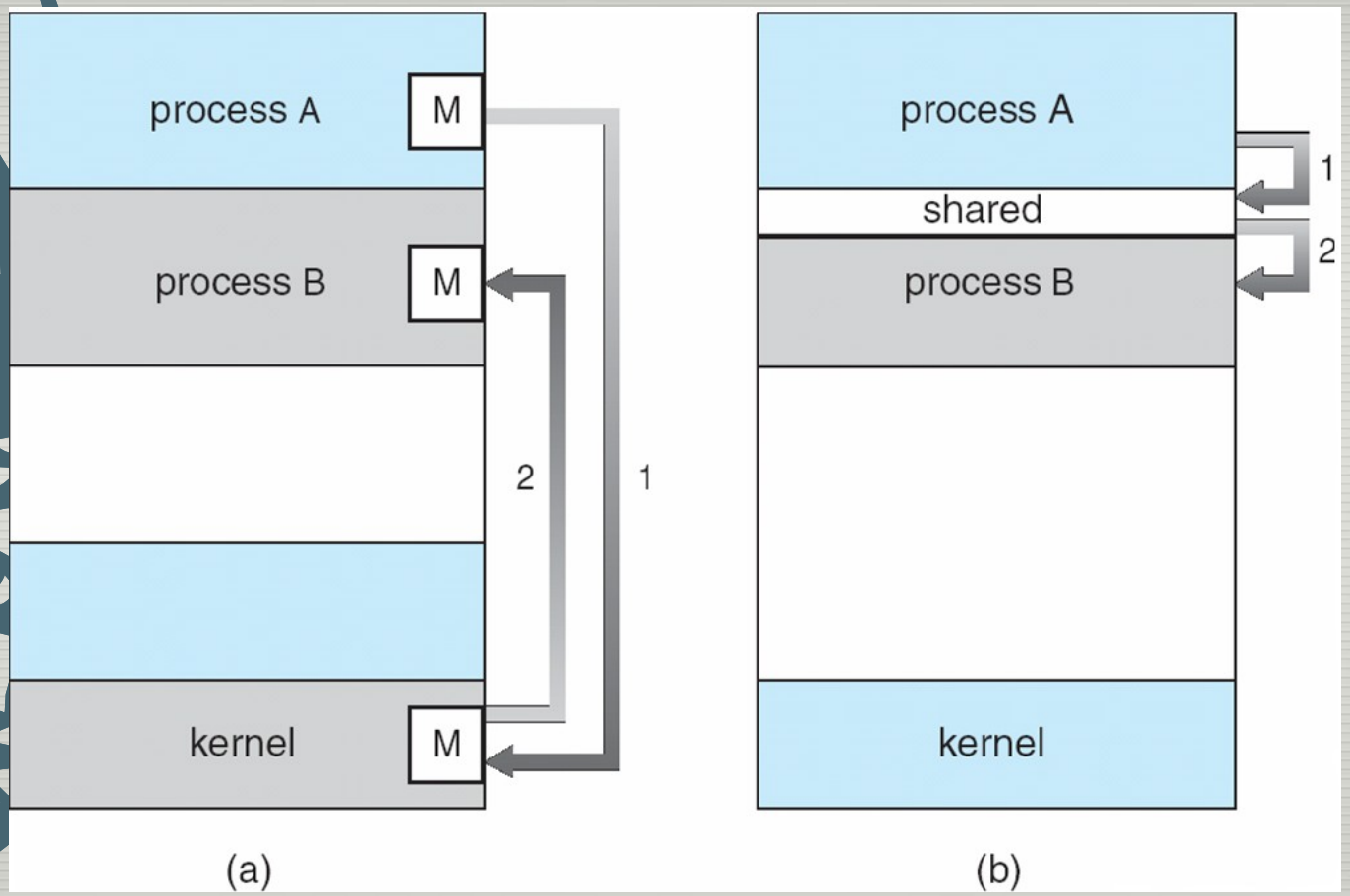
Inter-Process Communication(IPC)

- Processes can execute concurrently.
- **Independent process** : A process that can not affect of be affected by the other processes executing in the system.
- **Cooperating process** : A process that can affect or be affected by the other processes in the system.
- Reason for providing environment that allows process cooperation : Information sharing / Computation speedup / Modularity / Convenience.

IPC Mechanism

- A mechanism that allows cooperating processes to exchange data and information.
- **Shared Memory** & **Message Passing**

Message Passing & Shared Memory



Shared Memory

- Shared memory mechanism provides a region of shared memory.
- Other processes that wish to communicate must attach it to their address space.

Producer-Consumer Problem

- A producer process produces information.
- A consumer process consumes the information.
- A common paradigm for cooperating process.
- **One solution, using shared memory** : bounded buffer, unbounded buffer

Bounded Buffer Solution – Using Shared Memory

- Shared data

```
#define BUFFER_SIZE 10
typedef struct {
   . . .
} item;

item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

- Solution is correct, but can only use BUFFER_SIZE-1 elements

Bounded Buffer – A Producer

```
while (true) {
   /* Produce an item */
    while (((in = (in + 1) % BUFFER SIZE count)  == out)
     ;   /* do nothing -- no free buffers */
    buffer[in] = item;
    in = (in + 1) % BUFFER SIZE;
}
```

Bounded Buffer – A Consumer

```
while (true) {
    while (in == out)
        ; // do nothing -- nothing to consume

    // remove an item from the buffer
    item = buffer[out];
    out = (out + 1) % BUFFER SIZE;
return item;
}
```

Message-Passing Systems

- A mechanism for processes to communicate and to synchronize their actions.
- Provides at least two operations : send(message) / receive(message).
- IF processes P & Q want to communicate by sending and receiving the message, they need to
  - Establish a communication link between them
  - Send() / Receive() operations.

Methods for Implementing a Link

- Direct or indirect communication.
- Synchronous or asynchronous communication.
- Automatic or explicit buffering.

Direct communication

- Each processes must explicitly name the recipient or sender.
- send(P, message) / receive(Q, message) : Symmetry
- send(P, message) / receive(id, message) : Asymmetry
- Link properties
  - A link is established automatically.
  - A link is associated with exactly two processes.
  - Between each pair of processes, there exists exactly one link.

Indirect Communication

- The messages are sent to and received from a mailboxes(or port). – A abstract object
- Processes can communicate only if they share the mailbox.
- send(A, message) / receive(A, message)
- Link Properties
  - A link is established only if both members of the pair have a shared mailbox.
  - A link may be associated with more than two processes.

Indirect Communication

- Operations
  - Create a new mailbox.
  - Send and Receive messages through mailbox.
  - Destroy a mailbox.
- Mailbox Sharing
  - P_1, P_2, and P_3 share mailbox A
  - P_1, sends; P_2 and P_3 receive
  - Who gets the message?
- Solutions
  - Allow a link to be associated with at most two processes
  - Allow only one process at a time to execute a receive operation
  - Allow the system to select arbitrarily the receiver. Sender is notified who the receiver was.

Synchronization

- Message passing may be either blocking or non-blocking
- Blocking is considered synchronous
  - Blocking send has the sender block until the message is received
  - Blocking receive has the receiver block until a message is available
- Non-blocking is considered asynchronous
  - Non-blocking send has the sender send the message and continue
  - Non-blocking receive has the receiver receive a valid message or null
- Rendezvous between the sender and receiver is needed.

Buffering

- Queue of messages attached to the link; implemented in one of three ways
  - Zero capacity – 0 messages
    Sender must wait for receiver (rendezvous)
  - Bounded capacity – finite length of *n* messages
    Sender must wait if link full
  - Unbounded capacity – infinite length
    Sender never waits

Communication in Client-Server System

Sockets

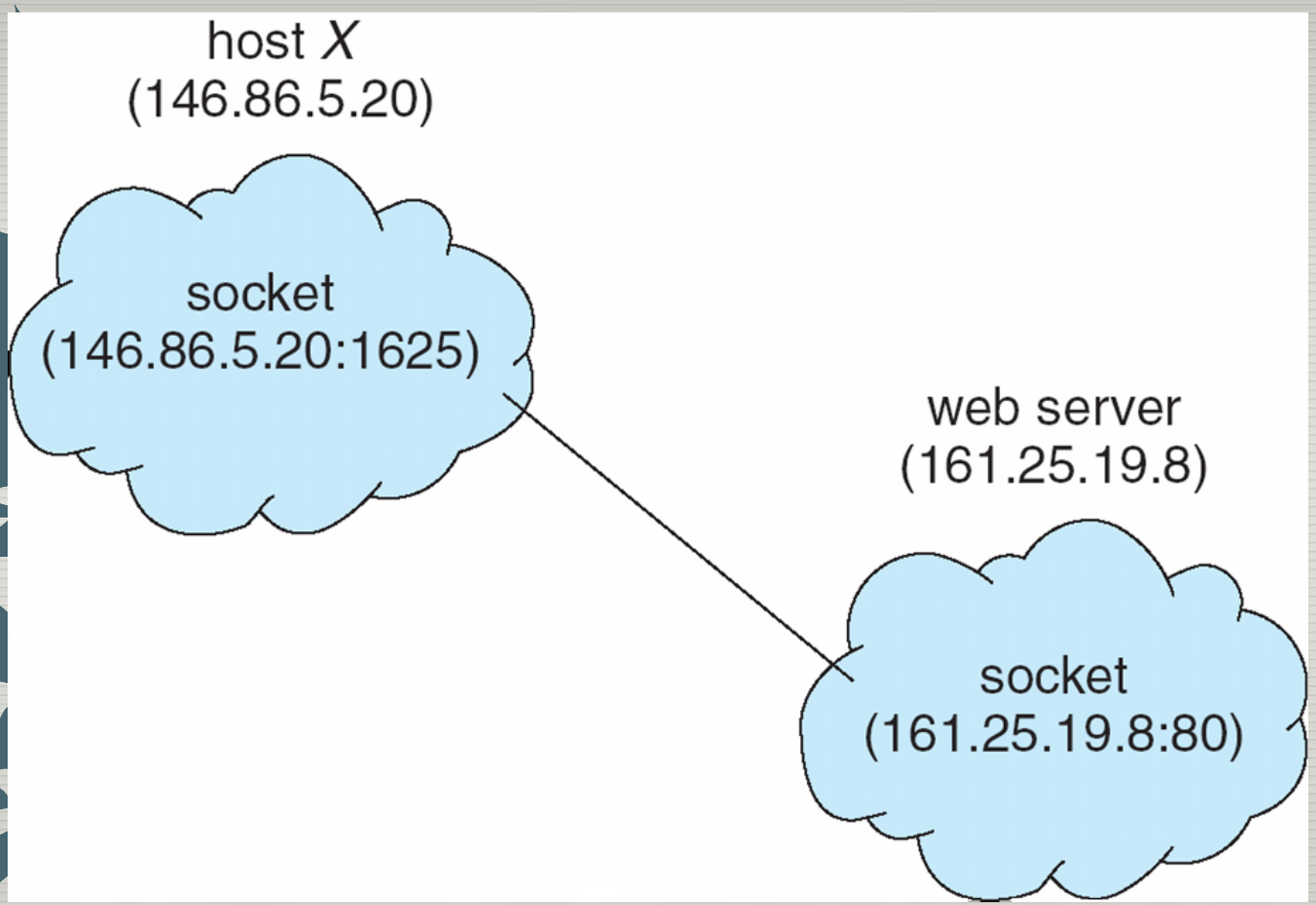
Remote Procedure Calls

Pipes

Sockets

- Definition : An endpoint for communication.
- A pair of processes communicating over a network employ a pair of sockets.
- A socket is identified by an IP address concatenated with a port number.
- Example : 18.18.26.5 : 1234

Socket Communitation



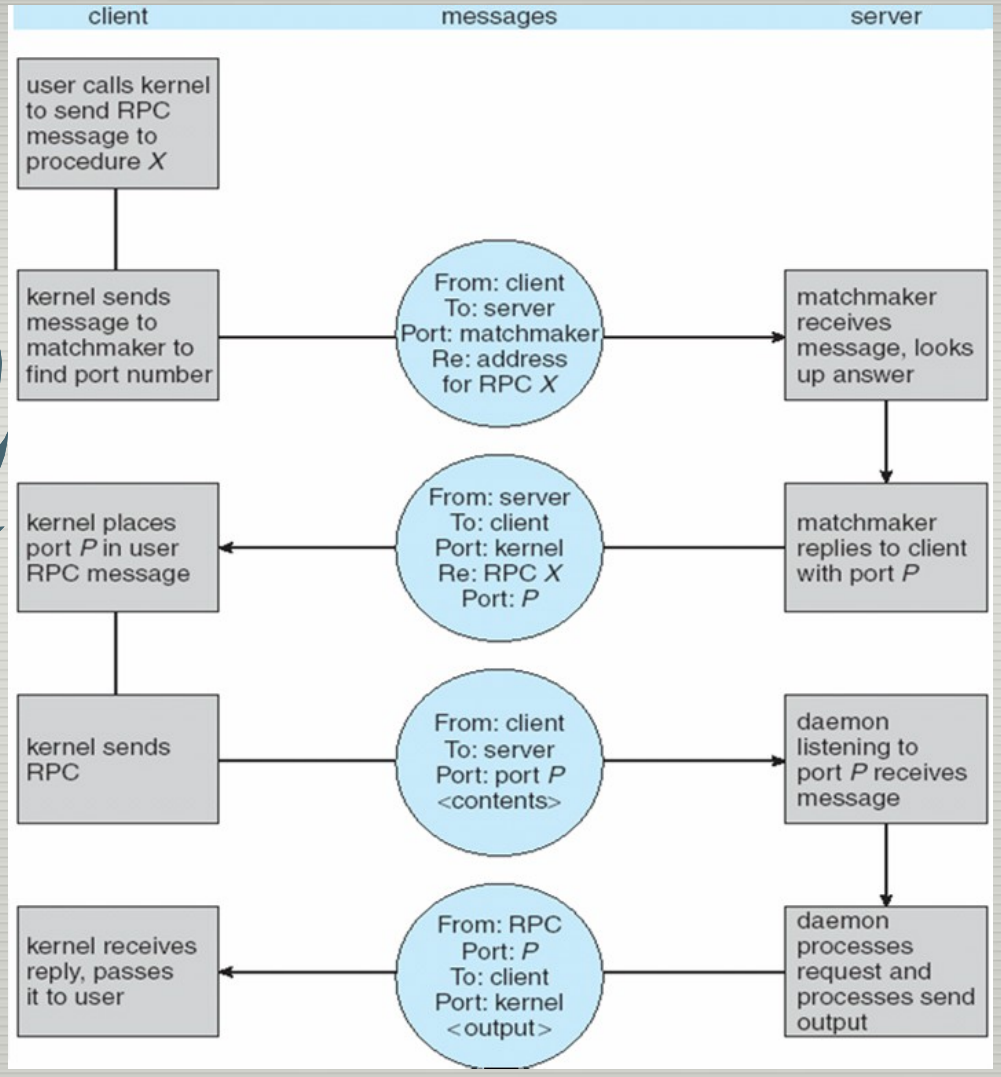
Remote Procedure Calls

- Remote procedure call (RPC) abstracts procedure calls between processes on networked systems
- The RPC system hides the details that allow communication to take place by providing a stub on the client side.
- The client-side stub locates the server and *marshalls* the parameters
- The server-side stub receives this message, unpacks the marshalled parameters, and performs the procedure on the server

External Data Representation(XDR)

- To adjust the machine dependencies, many RPC system uses machine-independent representation of data. – XDR
- Example : Network byte ordering.

Semantics of a Call



Pipes

- A pipe act as a conduit allowing two processes to communicate.
- Considering in implementing a pipe
  - Does the pipe allow unidirectional / bidirectional communication?
  - If two-way communication is allowed, is it half duplex or full duplex?
  - Must a relationship exist between the communication processes?
  - Can the pipes communicate over a network, or must the communicating process reside on the same machine?

Ordinary Pipes

- Anonymous pipe.
- Unidirectional pipe
- Two pipes exist : write/read-end
- Parent-Child Relationship is needed.

Named Pipe

- Exists as external object.(Ex : file)
- Relationship is not needed.
- Operations
  - Pipe creation.
  - Read and write operation.
  - Closing pipe.