Operating System

Process Management: Kernel in UNIX, Interprocess Communication-shared memory, message passing, client-server communication.

Session 6

Tutor: Yeshi Wangchuk, Asstt. Lecturer, IT Dept.

Kernel of Unix OS

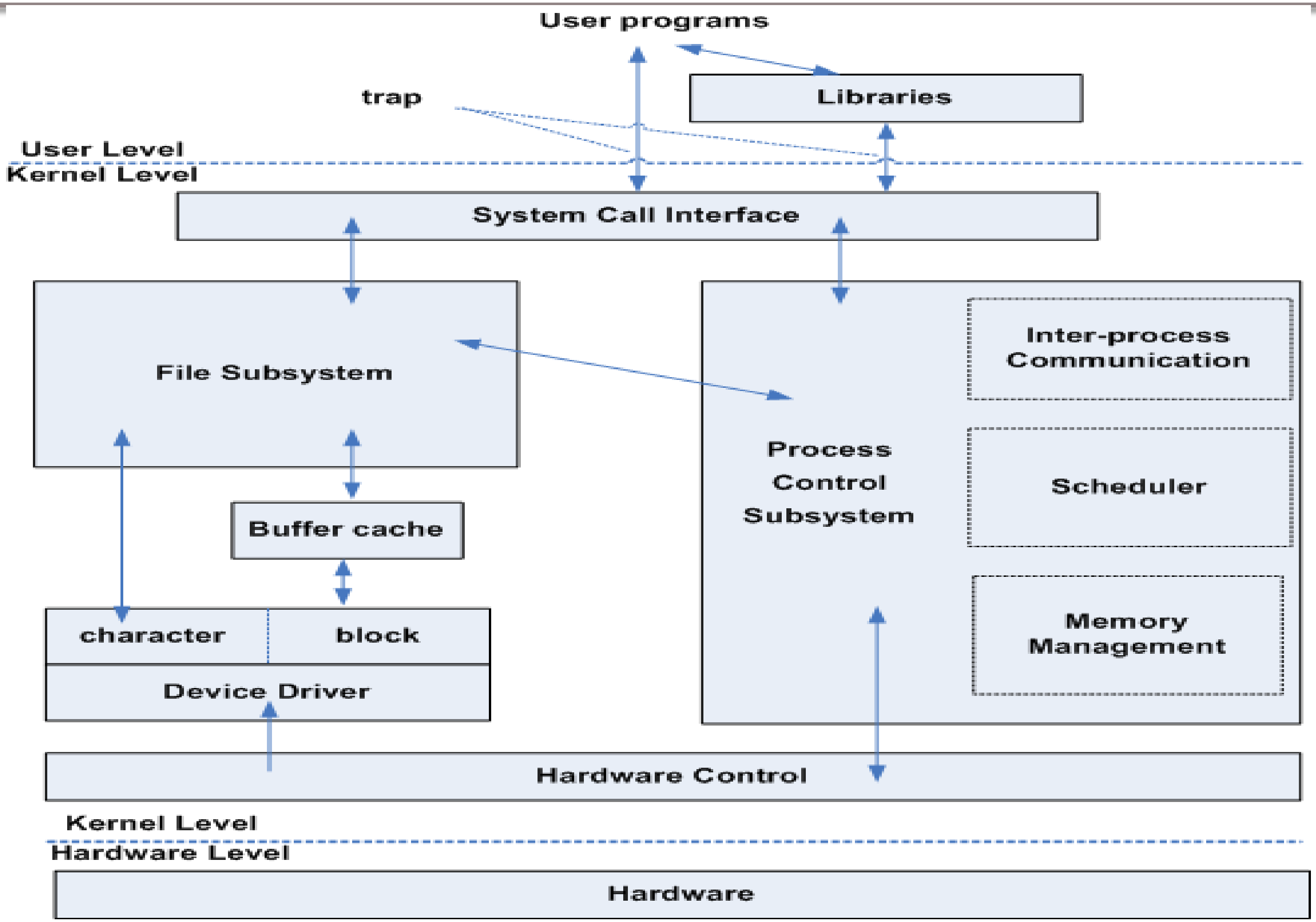


Figure 6.0: Block Diagram of the System Kernel

Interprocess Communication(IPC)

Introduction

- **IPC coordinates** b/w computation spreads over several processes.
- IPC enable the **communication** amongst processes & **synchronization** amongst processes.
- The needs for IPC arises from the **parallel and distributed context.**
- In **distributed environment**, IPC is useful where the ***communication processes may reside on different computers connected with a network***. Eg. **Chat program** used in WWW.

IPC at the Process Level

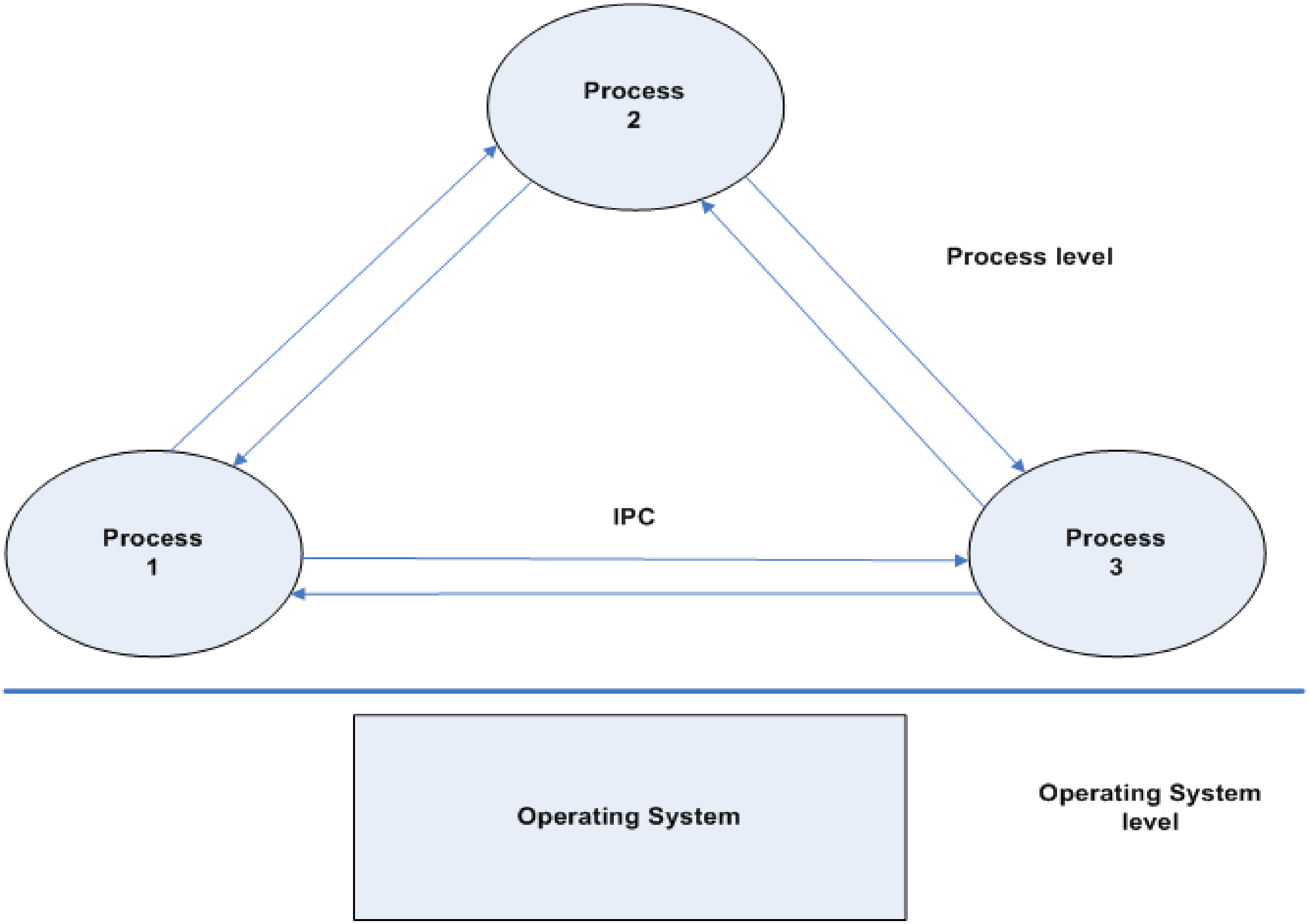


Figure 6.1: IPC at the process level

Fundamental Model of IPC

- There are two fundamental Model of Interprocess Communication:

1. **Shared Memory:** Memory is shared by cooperating processes.
2. **Message Passing:** Messages exchange b/w the cooperating processes.

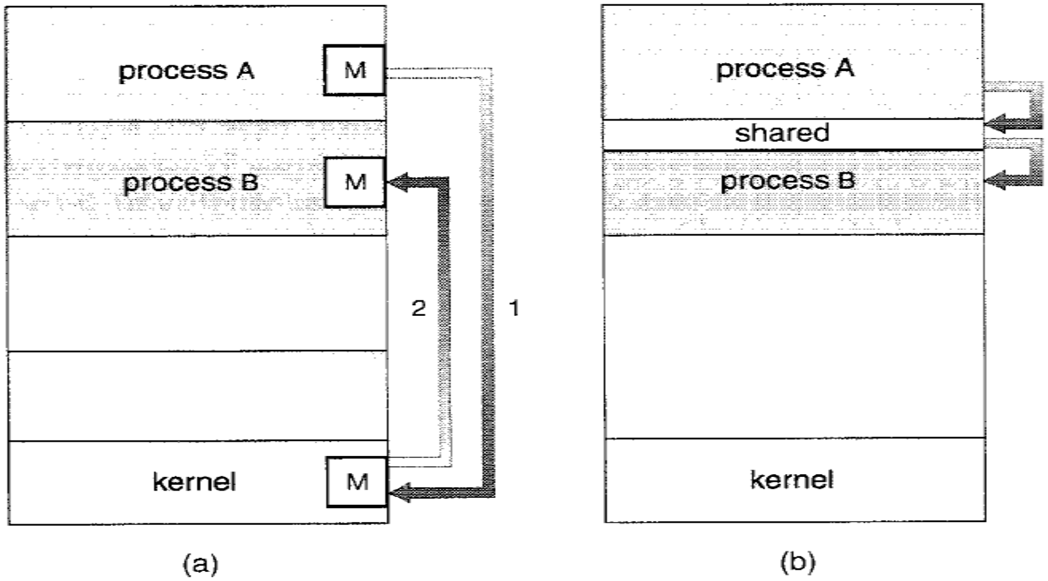


Figure 6.2: Communication Model, (a) Message passing. (b) Shared Memory

Shared Memory System -1

- ***Shared-memory region*** resides in the ***address space*** of the process creating ***shared-memory segment***.
- Other processes which needs to use shared-memory must be attached to address space.
- Example: ***producer-consumer problem***.
- Producer produces the items and consumer consumes the items.
- The ***solution to producer-consumer problem is shared-memory***.
- The producer and consumer must be ***synchronized***.
- Consumer waits if ***buffered is empty*** and producer wait if ***buffered is full.***

Shared **Memory System -2**

- Buffered of two types; ***unbounded and bounded buffered.***

```
#define BUFF_SIZE 10

typedef  struct{
      …..
}item;

item buffer[BUFF_SIZE];
int in;
int out;
```

- variable `in` point to the next free position in the buffer & `out` point to the first full position in the buffer.
- Buffer empty/full. When?
  - ✓ if(in==out)   //empty
  - ✓ if((in+1)%BUFF_SIZE==out) //Full.

Producer – Consumer Problem

```
//producer produces items;

item nextProduceItem;

while(true){
       while((in+1)%BUFF_SIZE==out)
              ;//do nothing
       buffer[in]=nextProduceItem;
       in=in+1%BUFF_SIZE;
}


//consumer consumes items;

item nextConsumeItem;

while(true){
       while(in==out)
           ;//do nothing
       nextConsumeItem=buffer[out];
       out=(out+1)%BUFF_SIZE;
}
```

Message Passing -1

- Message passing mechanism allows the processes to communicate and synchronize their action without using same address space.
- It is particularly useful in distributed environment(Eg. Chat)
- It provides at least two operations:
  - ✓ `send`(message) & `receive`(message)
- send()/receive() Operations:
  - ✓Direct or Indirect communication
  - ✓Synchronous or asynchronous communication
  - ✓Automatic or explicitly Buffering

Message Passing -2

- ***Direct or Indirect communication***
  - ✓ Direct:

    P: rend(Q,message) …..send message to process Q
    Q: receive(P,message) ….receive message from P

  - ✓ Indirect:

    Sends or receives from **Mailbox or ports**.
    P: rend(QMailBox,message) …..send message to process Q
    Q: receive(QMailBox,message) ….receive message from P

- ***Synchronous or asynchronous communication***
  - ✓ Synchronous(blocking): block until message is received/sent.
  - ✓ Asynchronous(nonblocking): continue sending or receiving whether received/sent or not.
- ***Automatic or explicitly Buffering***
  - ✓ Zero capacity: synchronous nature. Queue has max. length of 0.

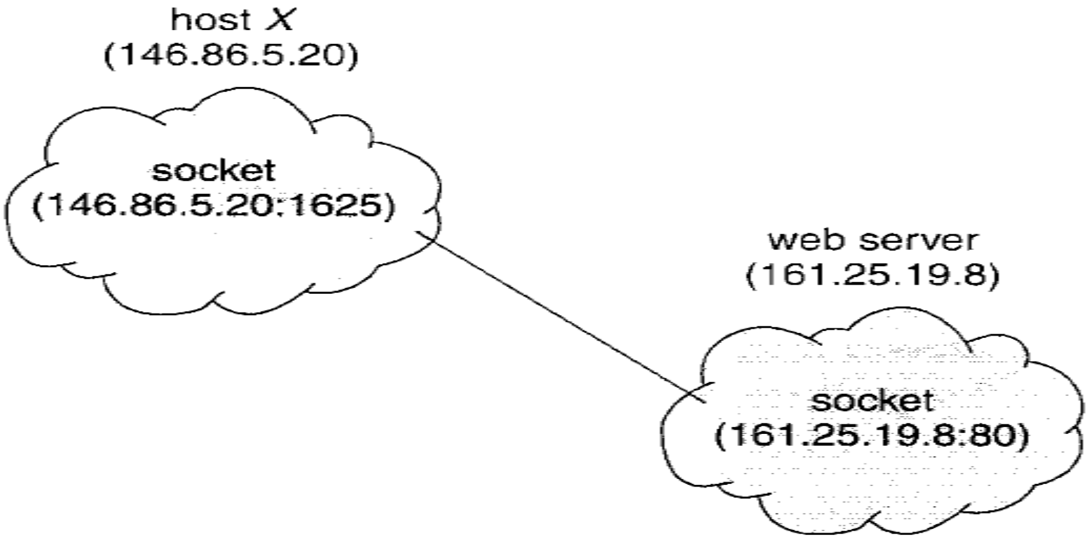
    Message system with no buffering.

  - ✓ Bounded capacity: Definite queue length.
  - ✓ Unbounded Capacity: Queue length is indefinite

  } System with automatic buffering

Comm. in Client-Server System

- Client request the server for services.
- Server grant the services to the requested clients.
- Communication using sockets.


host *X*
(146.86.5.20)
socket
(146.86.5.20:1625)
web server
(161.25.19.8)
socket
(161.25.19.8:80)