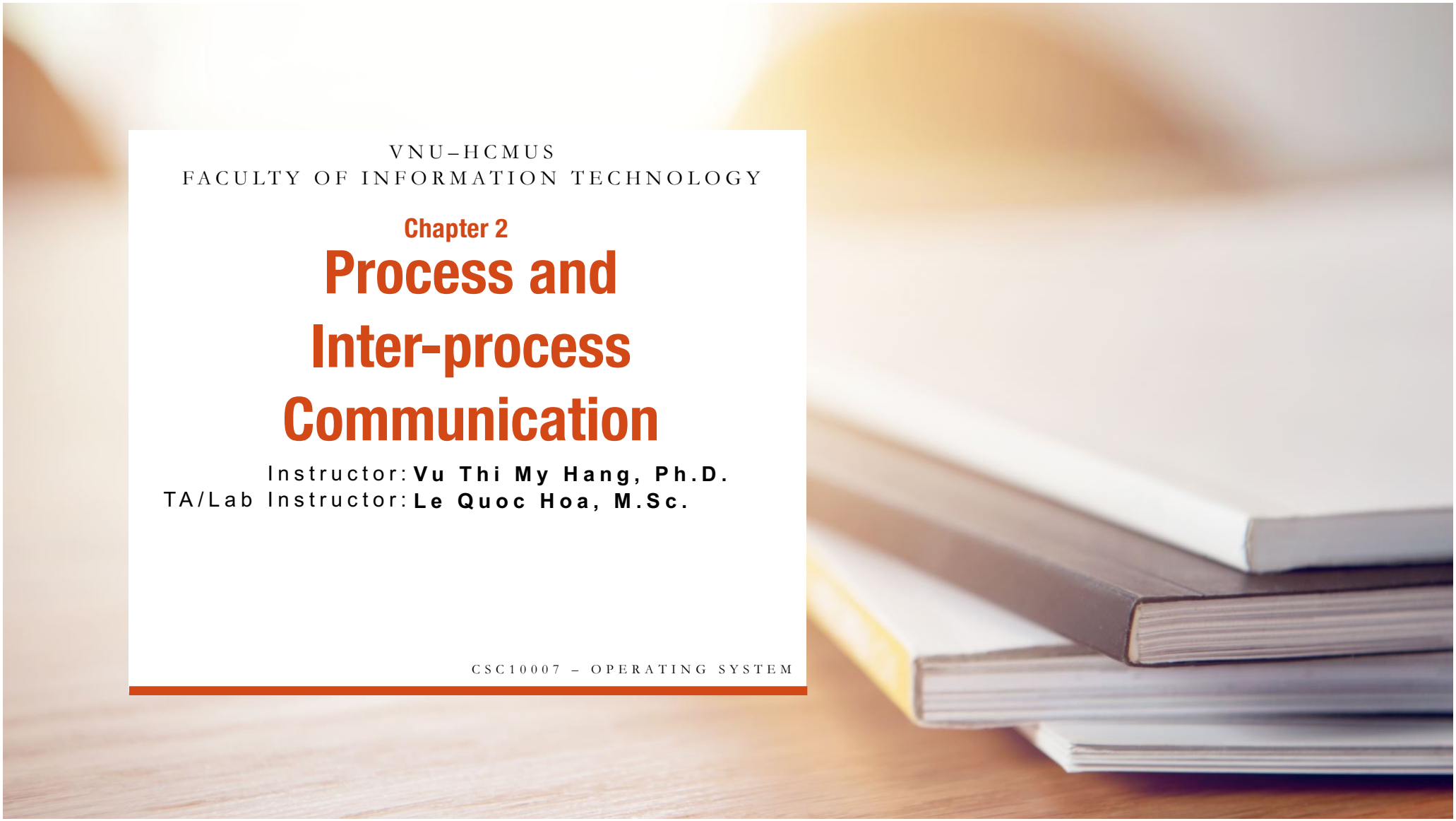


VNU–HCMUS
FACULTY OF INFORMATION TECHNOLOGY

Chapter 2

Process and Inter-process Communication

Instructor: **Vu Thi My Hang, Ph.D.**
TA/Lab Instructor: **Le Quoc Hoa, M.Sc.**

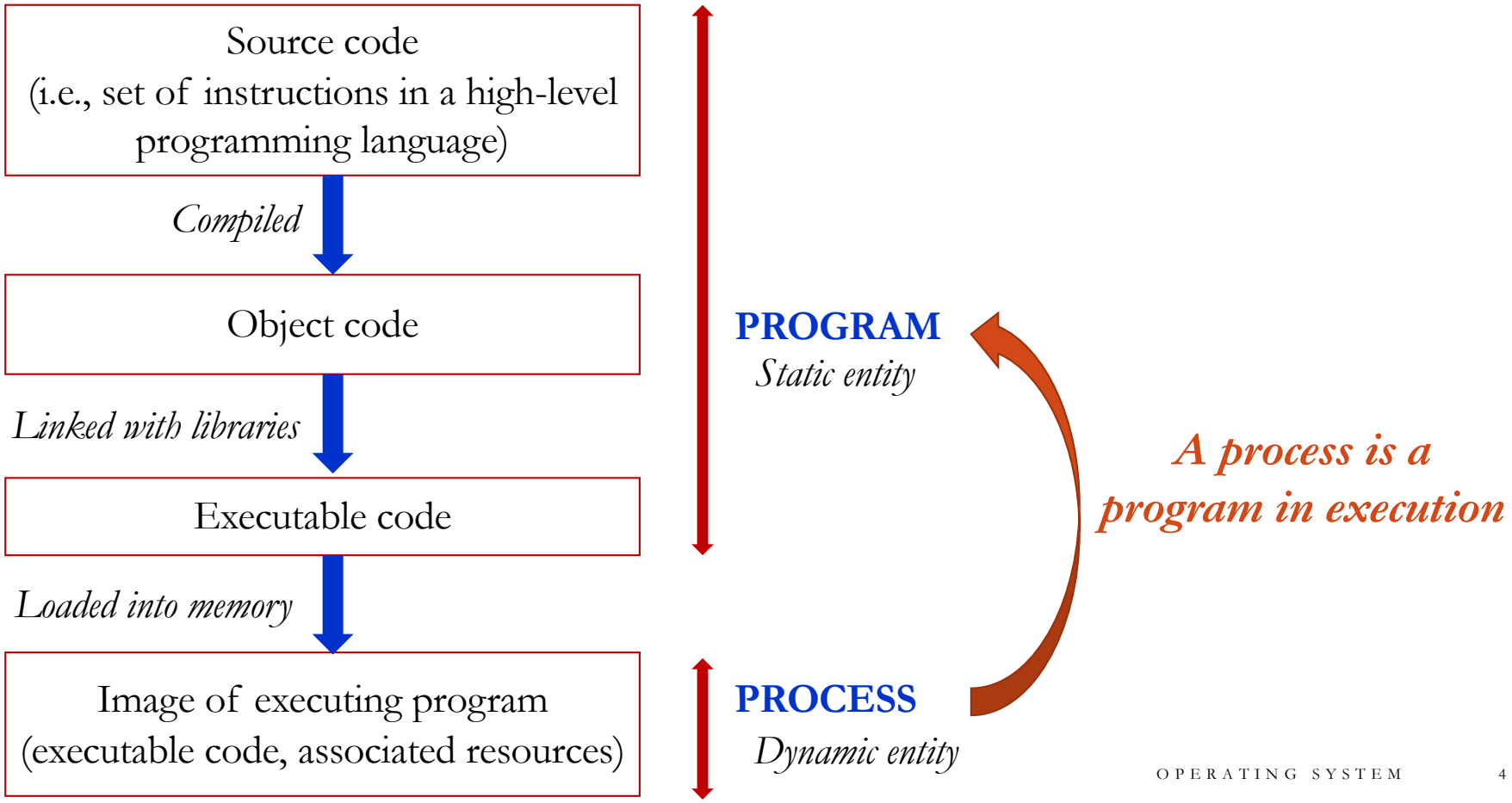
Plan

- Process concepts
- Process features
- Process operations
- Inter-process Communication (IPC)

Plan

- **Process concepts**
- Process features
- Process operations
- Inter-process Communication (IPC)

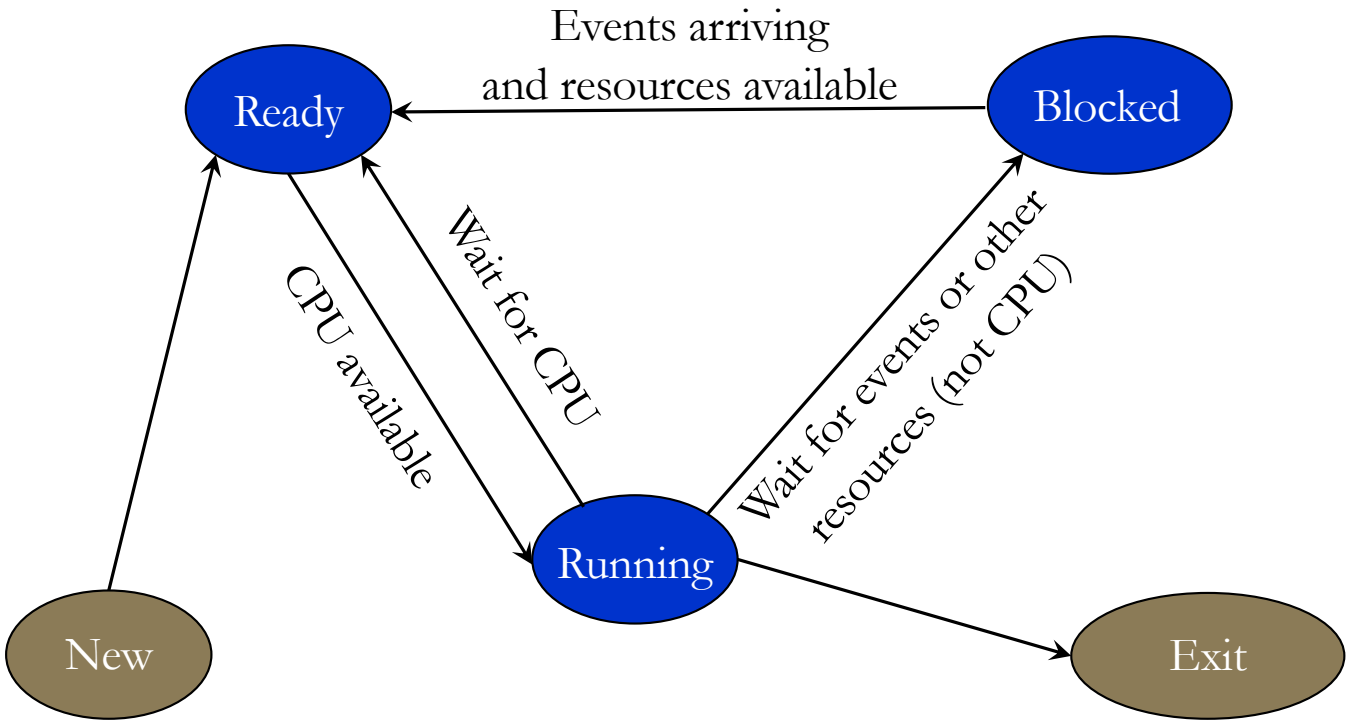
Process vs. Program (Tiến trình và Chương trình)



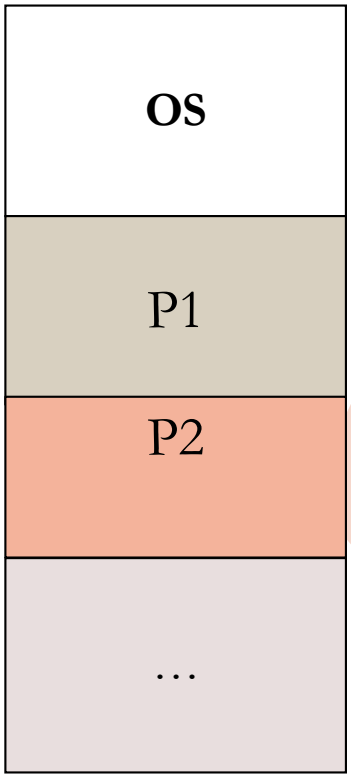
Plan

- Process concepts
- **Process features**
- Process operations
- Inter-process Communication (IPC)

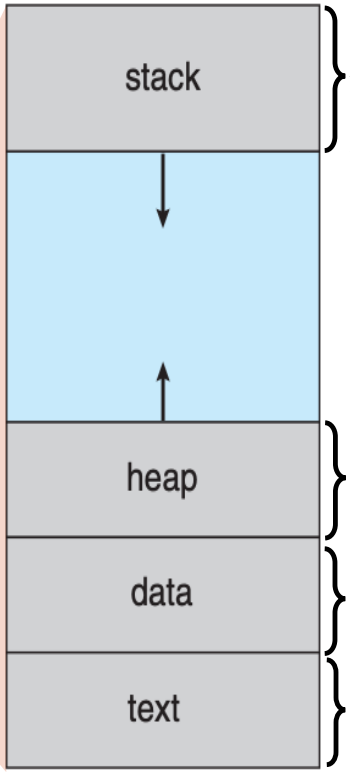
Process state (Trạng thái của tiến trình)



Process address space (Không gian địa chỉ của tiến trình)



Memory partitions



Address space reserved for each process

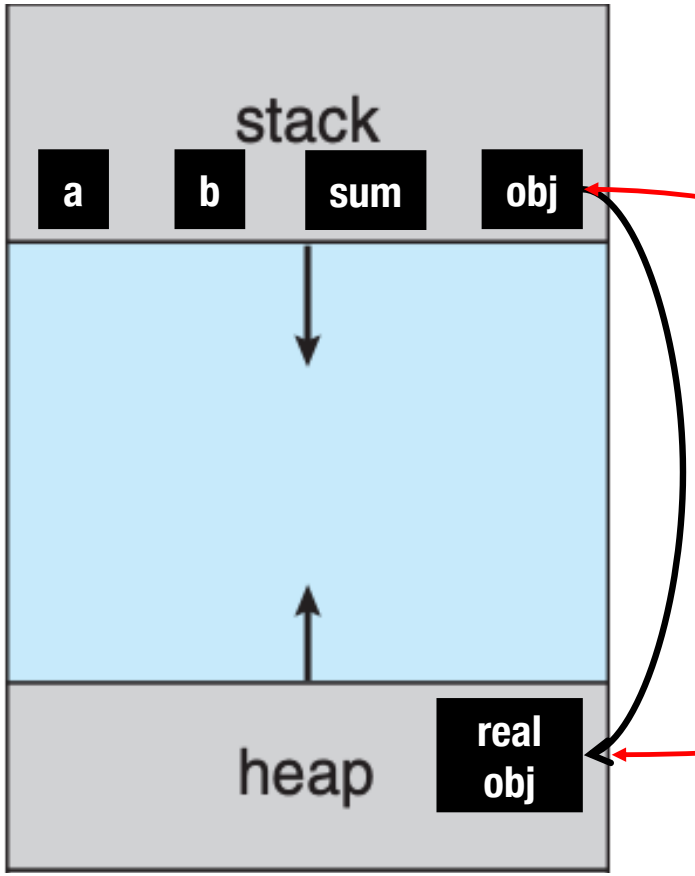
Function parameters, return values and local variables allocated automatically as a process loaded into memory

Variables allocated **dynamically** during process run time (e.g., objects, pointers)

Global (and/or static) variables

Program code (i.e., set of instructions)

Process address space (Không gian địa chỉ của tiến trình)



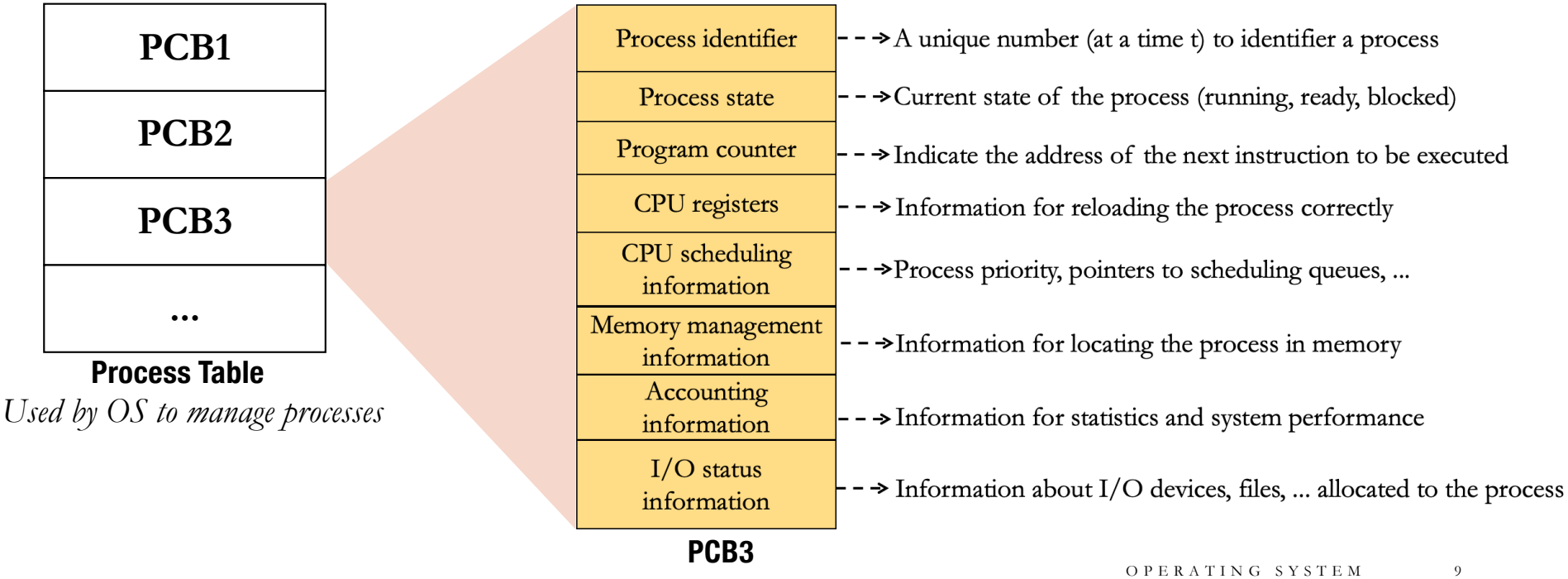

```
public class Addition {
        public int add(int a, int b) {
                int sum = a + b;
                return sum;
        }
}

void main() {
        Addition obj = new Addition();
        printf(obj→add(12, 25));
}
```

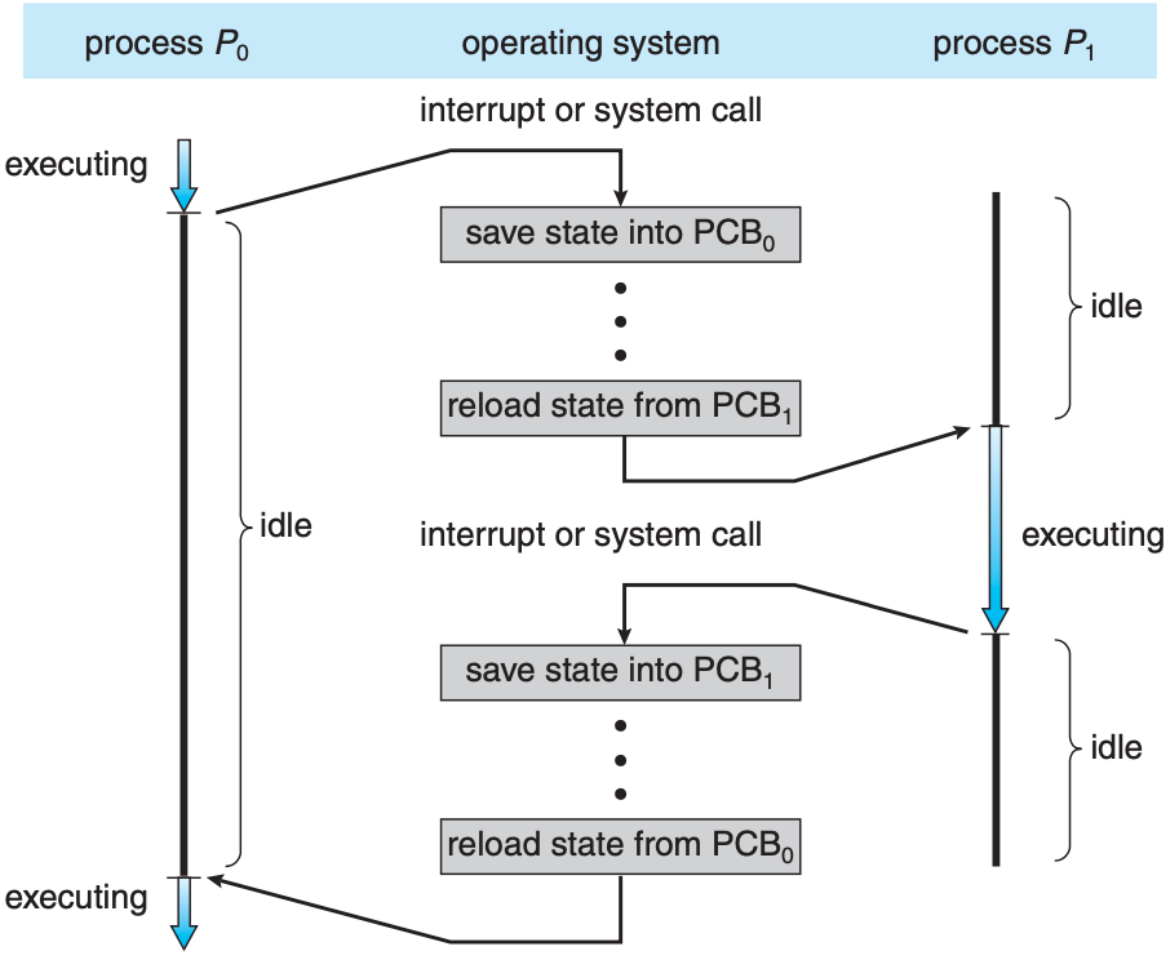
Where to find a, b, and sum in memory?
What about obj?

Process Control Block (PCB) (Khối quản lý tiến trình)

- Specific data structure
  - ✓ Contains process information
  - ✓ Located in OS address space and accessible only in kernel mode



Context Switch (Chuyển đổi ngữ cảnh)



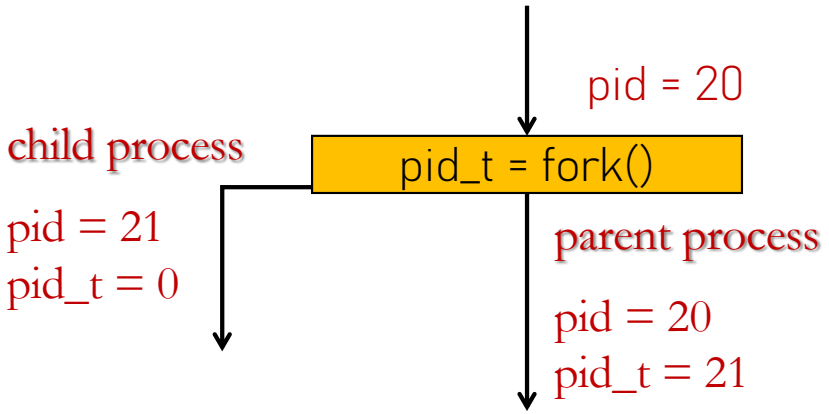
Plan

- Process concepts
- Process features
- **Process operations**
- Inter-process Communication (IPC)

Process creation

- A parent process create children processes, which, in turn, may create their own children processes
  - ✓ *System calls: fork (Unix), CreateProcess (Windows)*
- A child process may:
  - ✓ be a duplicate of its parent (i.e., have same memory space with its parent)
  - ✓ load another program to execute
    - *System call: exec (Unix)*
- When a new process is created:
  - ✓ It will be allocated a process identifier (*pid*) and other resources
  - ✓ A PCB containing the information of new process will be created and placed in Process Table

Process creation (Tạo lập tiến trình)




```
#include <unistd.h>
#include <stdio.h>

main( ){

        int pid_t = fork( );

        if ( pid_t ==0) { //child execution
                printf("%d is a child of parent %d", getpid(), getppid());
        }
        else if ( pid_t > 0) { //parent execution
                printf("Parent %d created a child %d", getpid(), pid_t);
            }
            else { error }

}
```

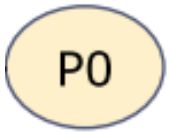
print 21 20

print 20 21

Process creation

Process Tree (Cây tiến trình)

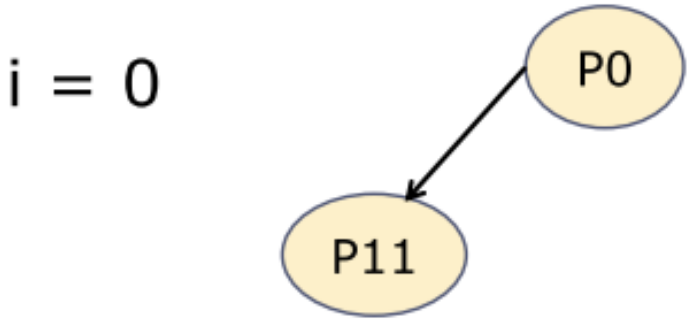
```
for(int i = 0; i < 3; i++){ fork();}
```



Process creation

Process Tree

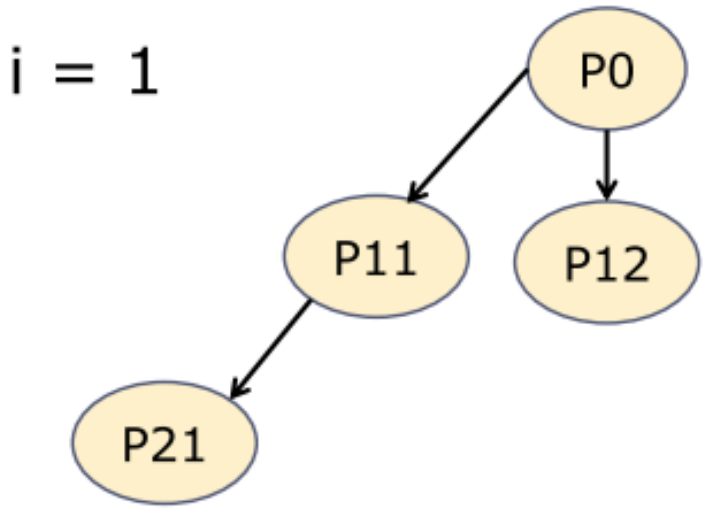
```
for(int i = 0; i < 3; i++){ fork();}
```



Process creation

Process Tree

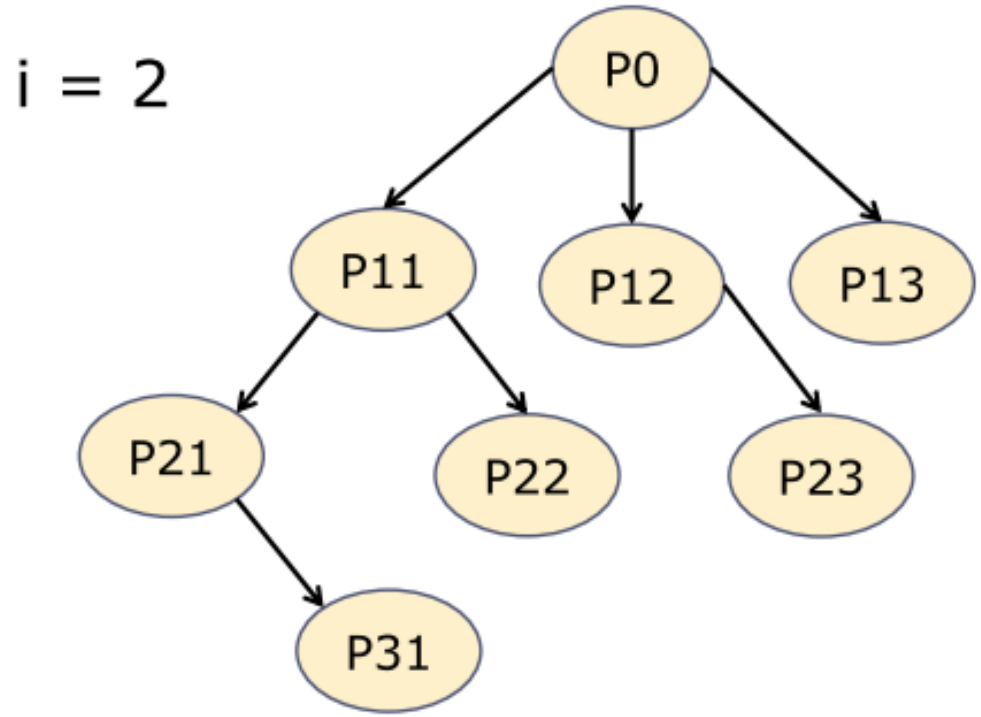
```
for(int i = 0; i < 3; i++){ fork();}
```



Process creation

Process Tree

```
for(int i = 0; i < 3; i++){ fork();}
```



Process termination (Kết thúc tiến trình)

- A process may terminate when finishing its execution or be terminated by its parent or OS processes
  - ✓ *System call: exit (Unix), ExitProcess (Windows)*
- Zombie processes
  - ✓ A child terminates, but its PCB has not released yet
    - → The parent must "wait" for its children to complete
      - *System call: wait (Unix), WaitForSingleObject (Windows)*
- Orphan processes
  - ✓ Parent terminates, but children still exist
    - → orphan processes → assigned to "root process"
  - ✓ Cascading termination

System Calls for Process Management

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()

Linux example

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

   /* fork a child process */
   pid = fork();

   if (pid < 0) { /* error occurred */
      fprintf(stderr, "Fork Failed");
      return 1;
   }
   else if (pid == 0) { /* child process */
      execlp("/bin/ls","ls",NULL);
   }
   else { /* parent process */
      /* parent will wait for the child to complete */
      wait(NULL);
      printf("Child Complete");
   }

   return 0;
}
```

Windows example

```
#include <stdio.h>
#include <windows.h>

int main(VOID)
{
STARTUPINFO si;
PROCESS_INFORMATION pi;

   /* allocate memory */
   ZeroMemory(&si, sizeof(si));
   si.cb = sizeof(si);
   ZeroMemory(&pi, sizeof(pi));

   /* create child process */
   if (!CreateProcess(NULL, /* use command line */
    "C:\\WINDOWS\\system32\\mspaint.exe", /* command */
    NULL, /* don't inherit process handle */
    NULL, /* don't inherit thread handle */
    FALSE, /* disable handle inheritance */
    0, /* no creation flags */
    NULL, /* use parent's environment block */
    NULL, /* use parent's existing directory */
    &si,
    &pi))
   {
      fprintf(stderr, "Create Process Failed");
      return -1;
   }
   /* parent will wait for the child to complete */
   WaitForSingleObject(pi.hProcess, INFINITE);
   printf("Child Complete");

   /* close handles */
   CloseHandle(pi.hProcess);
   CloseHandle(pi.hThread);
}
```

Plan

- Process concepts
- Process features
- Process operations
- **Inter-process Communication (IPC)**

Overview

(Cơ chế liên lạc/giao tiếp giữa các tiến trình)

IPC provides the way in which processes communicate to each other.

Exchange/Sharing data

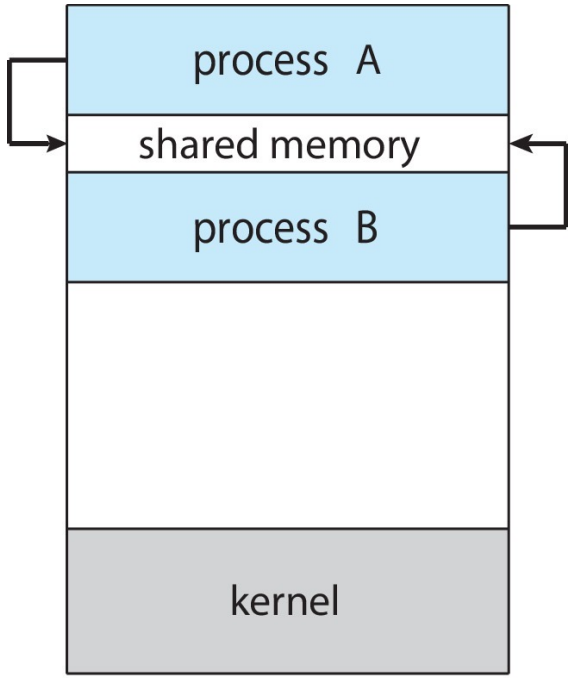
Cooperate to accomplish a task

Pipe (Đường ống)

- Unidirectional communication
  - ✓ Sending data in a one-way direction
- Consists of two processes: writer and reader
- Two kinds of pipes
  - ✓ Ordinary pipes (i.e., anonymous pipes): two communicating processes must have a parent-child relationship
  - ✓ Named pipes: two communicating processes do not need to have a parent-child relationship

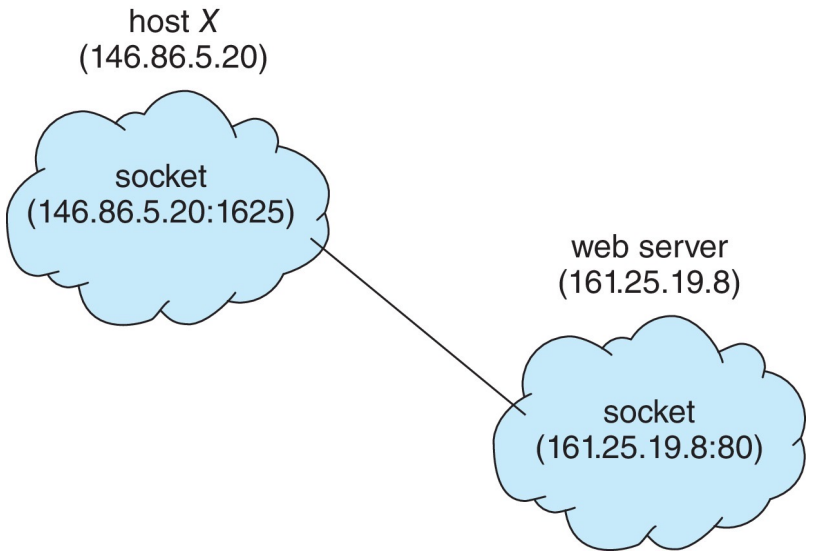
Shared Memory (Vùng nhớ chia sẻ)

- Processes make communication via a shared memory
  - ✓ Shared memory resides in address space of the process that created it
  - ✓ Generally, memory sharing requires a synchronization mechanism



Socket

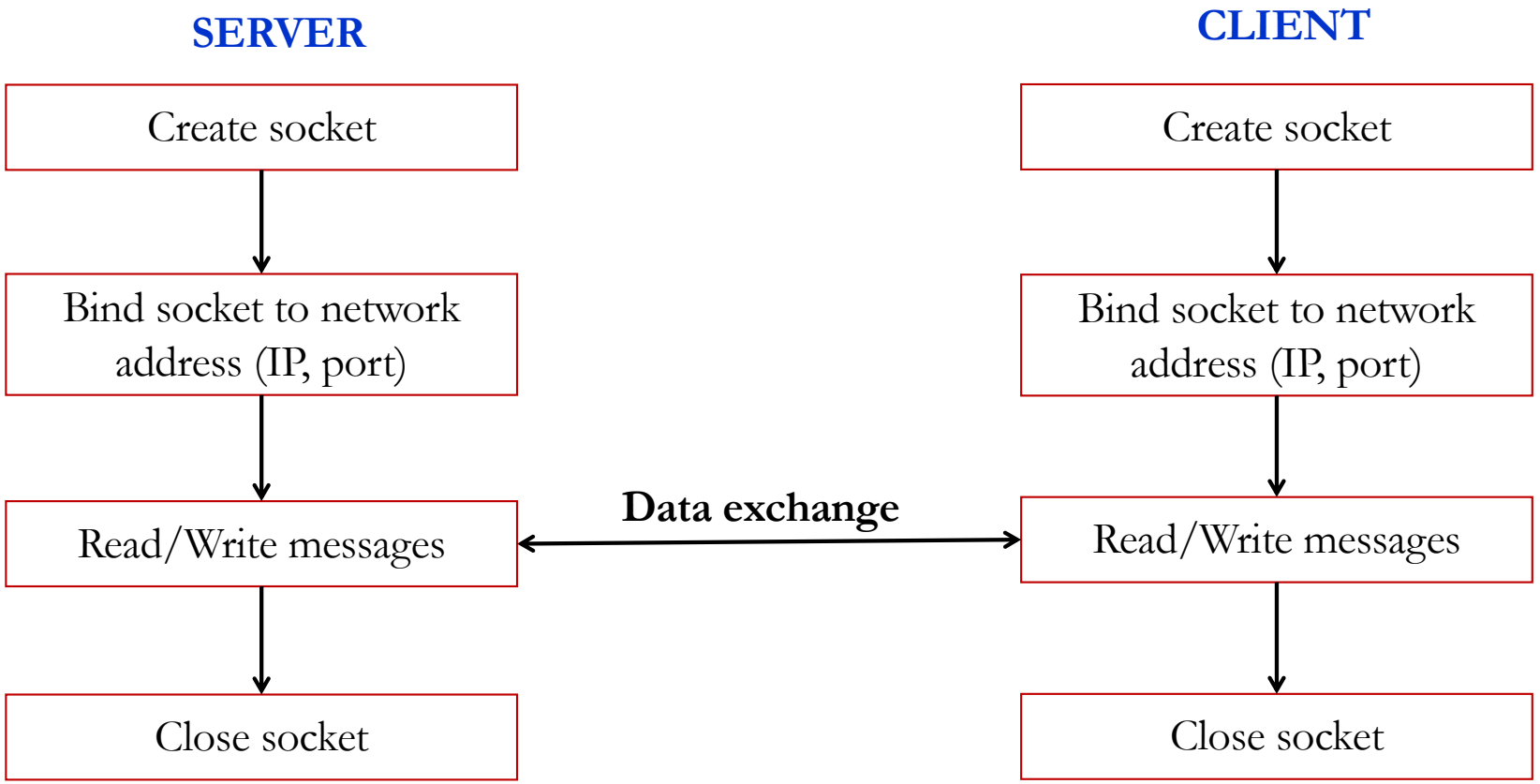
- Communication endpoints between processes on distant machines
  - ✓ Based on client-server model
- A socket is identified by an **IP address** and a **port number**
  - ✓ IP address: computer identifier over network
  - ✓ Port number: program identifier on a computer



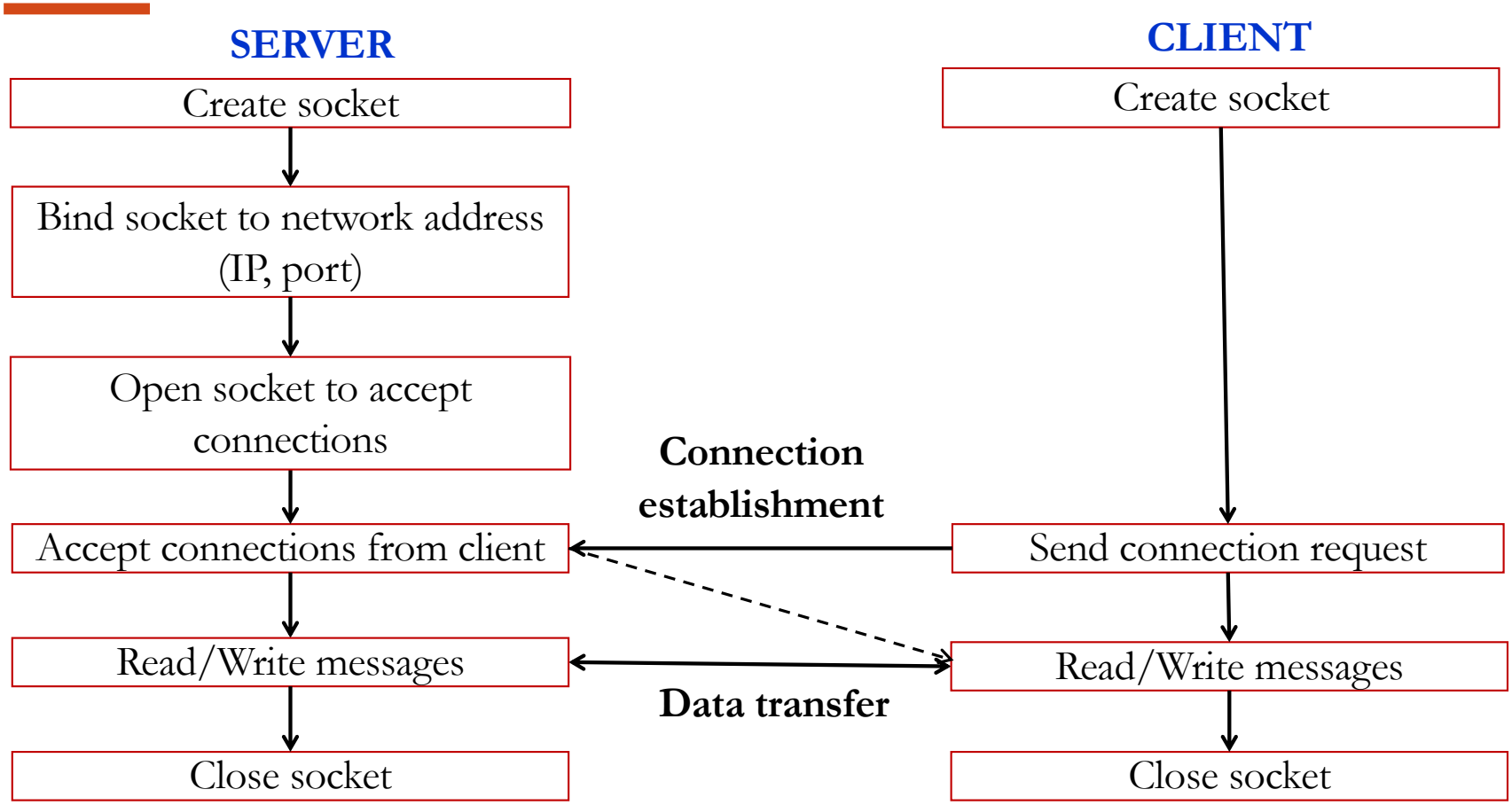
Socket

- Connectionless (UDP) socket
  - ✓ No connection required
  - ✓ Data transmitted as datagrams
  - ✓ Loss data possible
  - ✓ Fast
  - ✓ Less reliable than TCP socket
- Connection-oriented (TCP) socket
  - ✓ A logical connection required during data transfer
  - ✓ Data transmitted as streams of bytes
  - ✓ Re transmission
  - ✓ Slower than UDP socket
  - ✓ Reliable

Socket: Connectionless (UDP) socket



Socket: Connection-oriented (TCP) socket



And other ...

- Signal handling
- RCP (Remote Control Protocol)
- Message Passing