University of Engineering and Technology, Lahore
Operating Systems Lab.

Semester: 6th **Session: 2019-23**

Lab-05 (Individual)

Submission Date: -03-22 (Before 11:49 pm)

Note: Understanding the assignment is part of the assignment.

Inter Process Communication

In computing, inter-process communication (IPC) is a set of methods for the exchange of data among multiple processes. Processes may be running on one or more computers connected by a network.

Commonly used methods for IPC are:

1. File	Most operating systems
2. Pipe	All POSIX systems, Windows
3. Shared memory	All POSIX systems, Windows

The method of IPC used may vary based on the bandwidth and latency of communication between the threads, and the type of data being communicated.

There are several reasons for providing an environment that allows process cooperation:

1. Information session
2. Computational speedup
3. Modularity
4. Convenience

(Source: Wikipedia)

I showed you in classes that suppose we are working in a Microsoft word. Now the word is running as a separate process and has its own address space. For simplicity assume that address space is the RAM allocated to a process. No process can ever have access to another process's address space directly. So OS provides us different mechanism for this so that two or more processes can communicate with each other. Now if we try to insert a graphical chart in word document it will open a Microsoft excel document, which will contain the numerical data for the chart. From now on every change in numerical data in excel will be reflected in the word document. This shows that there is some sort of communication going on between the two processes. This is just a very basic example to show the mechanism.

Pipes:

Piping is a process where the output of one process is made the input of another and vice versa.

Task:
You will use three processes and make them communicate with each other via pipe. Write a C/C++ program in which the main process (parent process) will create 2 child processes and then both the child processes will send a STRING MESSAGE to its parent process. The message will just contain you roll number and name.

Gets Hands Dirty:

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

using namespace std;
int main(){

      int buffer[2];
      pid_t pid;

            cout << " Student RoLL NO and Name";

      if(pipe(buffer) < 0){

            cout << "\n\tPipe Function Failed!";
            return 0;


      }
//Your code here



      char* message = (char*)"Message From Child 1: ABC";

            cout << "\n\t1st Child Process ID: " << getpid() << "\n";

            write(buffer[1], message, strlen(message));

      } else {

            cout << "\n\tParent Process ID: " << getpid() << "\n";


//Your code here
```

```
                write(buffer[1], message, strlen(message));
            } else {
                char *message = (char*)malloc(55);
                wait(NULL);
//Your code here
      }
      return 0;
}
```