✅ **CODE EXPLANATION (Line-by-line)**

```
// IPC using PIPE (unidirectional: parent sends, child receives)
#include <stdio.h>
#include <unistd.h>
```

- stdio.h – for input/output functions like printf
- unistd.h – contains pipe(), fork(), read(), write()

```
int main() {
  int fd[2];        // fd[0] = read end, fd[1] = write end
  pipe(fd);         // Create pipe
```

- fd[0] → read end of pipe
- fd[1] → write end of pipe
- pipe(fd) creates a communication channel.
  - If successful, returns 0
  - Data written to fd[1] can be read from fd[0]

```
  int pid = fork();   // Create child process
```

- fork() creates a **child process**
- Returns:
  - > 0 → running in **parent**
  - 0 → running in **child**
  - < 0 → error

👨‍👦 **Parent Process Code**

```
  if (pid > 0) {      // Parent Process
    close(fd[0]);   // Close read end
    char msg[] = "Hello from Parent!";
    write(fd[1], msg, sizeof(msg));
    close(fd[1]);
  }
```

- Parent **does not read**, so close(fd[0])

- Writes the message "Hello from Parent!" to the pipe using write()
- sizeof(msg) ensures full string is sent including \0 (null terminator)
- Closes writing end after sending

👶 **Child Process Code**

```
else {          // Child Process
    close(fd[1]);   // Close write end
    char buffer[50];
    read(fd[0], buffer, sizeof(buffer));
    printf("Child received: %s\n", buffer);
    close(fd[0]);
}
```

- Child **does not write**, so close(fd[1])
- Reads data sent by parent into buffer
- Prints the received message
- Closes the reading end

```
    return 0;
}
```

🖥️ **OUTPUT**

Child received: Hello from Parent!

📌 **THEORY FOR PRACTICAL EXAM**

🔷 **What is IPC?**

IPC (Inter-Process Communication) allows processes to communicate and share data with each other.

🔷 **What is a Pipe?**

A **pipe** is a **unidirectional** communication channel used between related processes (like parent-child).

🔷 Types of Pipes

Type	Direction	Use
Unidirectional Pipe	One-way (Parent → Child OR Child → Parent)	Default in UNIX
Bidirectional Pipe	Two-way	Requires socket or two pipes

🔷 Properties of Pipe

- ✅ Works in **one direction**
- ✅ **Half-duplex** communication
- ✅ Exists **only in memory** (not file)
- ✅ Automatically deleted after program ends
- ✅ Requires **related processes** (via fork())

🔷 Pipe Ends

fd index	Meaning
fd[0]	Read end
fd[1]	Write end

🔁 ALGORITHM

1. Create a pipe using pipe(fd)
2. Call fork() to create child process
3. In **parent**:
    - Close read end (fd[0])
    - Write data to pipe (fd[1])
4. In **child**:
    - Close write end (fd[1])
    - Read data (fd[0])
5. Display received data
6. Close unused pipe ends

✅ ADVANTAGES

Advantage

Simple and fast way of IPC

No special permissions needed

Works well between parent-child processes

Efficient in memory (buffer-based)

❌ **DISADVANTAGES**

Disadvantage

Unidirectional (one-way)

Only works for related processes (via fork)

Limited buffer size

Cannot send complex data structures directly