

**Vidyavardhini’s College of Engineering & Technology**

Department of Computer Engineering Academic Year : 2024-25

|  |  |  |  |
| --- | --- | --- | --- |
| **Class:** | **BE** | **Semester:** | **VIII** |
| **Course Code:** | **CSL801** | **Course Name:** | **Distributed Computing Lab** |

|  |  |
| --- | --- |
| **Name of Student:** | **Pratima Dinkar Bombe** |
| **Roll No. :** | **07** |
| **Division:** | - |
| **Experiment No.:** | **03** |
| **Title of Experiment:** | **Communication using Pipes** |
| **Date of Submission:** | **28/01/2025** |
| **Date of Correction:** | **04/02/2025** |

Evaluation

|  |  |  |
| --- | --- | --- |
| **Performance Indicator** | **Max. Marks** | **Marks Obtained** |
| Performance | 5 |  |
| Understanding | 5 |  |
| Journal work and timely submission | 10 |  |
| Total | 20 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Performance Indicator** | **Exceed Expectations (EE)** | **Meet Expectations (ME)** | **Below Expectations (BE)** |
| Performance | 4-5 | 2-3 | 1 |
| Understanding | 4-5 | 2-3 | 1 |
| Journal work and timely submission | 8-10 | 5-8 | 1-4 |

**Checked by**

**Name of Faculty : Ms. Swati Varma**

**Signature :**

**Date :**

EXPERIMENT 3

**Aim:** To implement communication using Pipes

**Objective:** Develop a program to implement communication using Pipes

**Theory:**

Pipes are a method of IPC that allows data transfer between processes. They are commonly used for communication between a parent and a child process in Unix-like operating systems.

**Characteristics of Pipes**

* **Unidirectional**: Data flows in only one direction (one process writes, the other reads).
* **Half-Duplex Communication**: Since data flows in one direction, two pipes are required for bidirectional communication.
* **Related Processes Only**: Pipes are used for communication between processes that have a common ancestor (e.g., parent-child processes).
* **Automatic Synchronization**: Pipes ensure data is written and read in sequence, avoiding race conditions

**Steps for IPC Using Pipes**

1. **Create a pipe** using os.pipe().
2. **Fork a child process** using multiprocessing or fork().
3. **Close unused pipe ends** in each process:
   * In the **parent**, close the **read end**.
   * In the **child**, close the **write end**.
4. **Write data** to the pipe from one process.
5. **Read data** from the pipe in the other process.
6. **Close the pipe ends** after communication to free resources.

**Code and output**:

import os

import multiprocessing

def child\_process(pipe\_write):

    os.close(pipe\_write[0])  # Close unused read end

    message = "Hello from child process!"

    os.write(pipe\_write[1], message.encode())  # Write message to pipe

    os.close(pipe\_write[1])

def parent\_process():

    pipe\_read, pipe\_write = os.pipe()  # Create pipe

    # Create child process (pass pipe\_write as a tuple, not a list)

    process = multiprocessing.Process(target=child\_process, args=((pipe\_read, pipe\_write),))

    process.start()

    os.close(pipe\_write)  # Close unused write end

    # Read message from child process

    message = os.read(pipe\_read, 1024).decode()

    print(f"Parent received: {message}")

    os.close(pipe\_read)

    process.join()

if \_\_name\_\_ == "\_\_main\_\_":

    parent\_process()



**Conclusion:** The parent and child processes successfully communicated using the pipe mechanism, demonstrating inter-process communication (IPC). The child process wrote a message to the pipe's write end, while the parent process read the message from the pipe's read end. Proper handling of pipe ends ensured synchronization and prevented resource conflicts. The use of os.pipe() enabled unidirectional data flow, and closing unused pipe ends in each process ensured efficient resource management. This experiment highlights the effectiveness of pipes in IPC, particularly for communication between related processes like parent and child.