**IMPLEMENTATION OF INTER-PROCESS COMMUNICATION [IPC]**

**Team**

1. *Rishwanth Reddy Baddipadaga, 11515020*
2. *Amit Reddy Baddam, 11507588*
3. *Roja Kamble, 11454258*
4. *Pranavanath Reddy Jaggari, 11503083*

**Introduction**

**[a]** ***Overview of the project***

We are coming up with an idea to implement Inter process communication (IPC) through shared memory and message passing. To enable co-operation and establish communication between two processes, there’s a need for an efficient & simple communication mechanism. IPC exactly serves the same purpose. It allows processes to communicate with each other and co-operation is enabled.

We are implementing IPC through

* Shared Memory
* Message Passing
* Pipes

IPC through Shared Memory facilitates two or more processes to access the memory and communication is established through the same where one process can observe changes made by other process. IPC through message passing is achieved through either establishing a communication link or using an existing one to exchange messages through primitives like send and receive. Pipes are useful for communication between related processes i.e., Inter-Process Communication. They act as a one-way communication as one process write to it and the other reads from it. If in any case, a process tries to read before something is written to the pipe, the process is suspended until something is written.

**[b]** ***What problem being solved?***

* Communication between two connected processes. It has a half-duplex mechanism. In other words, the first process talks to the second process. Full duplex requires additional channel for communication between the second and first processes.
* Communication between two or more processes with full-duplex capability. Processes communicate with each other by sending messages and retrieving them from queues. Once retrieved, the message is no longer available in the queue.
* Shared memory must be protected from each other by synchronizing access to all processes.

**[c]** ***Why is it important?***

IPC plays a crucial role in how computer systems work. It makes it possible for various applications to operate concurrently, share information, and communicate with one another. IPC is essential for the smooth running of an operating system and makes sure that the tasks are completed correctly and in the correct sequence. IPC plays an important role in the design of micro- and nano-kernels, which limit the functionality offered by the kernel. When compared to a typical monolithic kernel, certain features are subsequently acquired by interacting with servers via IPC, which significantly increases communication. Modularity can be accelerated with the use of inter-process communication.

**Background**

**[a]** ***What does one need to know to understand the problem?***

One must understand the concept of process in OS, memory, threads, pipes, communication between multiple computers connected by a network.

Approach of IPC is as shown in below figure.

**Chart, bubble chart

Description automatically generated**

**Few Definitions:**

**IPC Definition:** Data exchange across different threads in one or more processes or programs occurs through inter-process communication.

**Pipe Definition:** For communication between two linked operations, pipe is frequently utilized.

**Message Passing Definition:** A process can synchronize and communicate through message forwarding.

**Message Queue Definition:** A linked list of messages kept in the kernel is referred to as a message queue.

**Shared Memory Definition:** Memory that is shared by two or more processes that is created utilizing shared memory for all processes is known as shared memory.

**Implementation**

**[a] *What are the solutions to your problem?***

* Implementing faster communication between processes, bidirectional using message queues or shared memory segment, communicate with some information from its own address space to other processes, using shared memory IPC method.
* Implementing multilevel communication, between parent, child, and grandchild, using pipes IPC method.
* Implementing message passing, Message passing model allows multiple processes to read and write data to the message queue without being connected to each other, using message passing IPC method. The message passing model is much easier to implement than the shared memory model.

**[b] *How do you implement?***

* Programming language: C
* Operating system: Mac OS/Windows
* Editor: VS Code, Terminal [CLI]
* Logic Hook: Algorithms

Understanding the algorithms and applying to build logic with proper analyzed inputs and verifying the expected output.

***i. Programming language for implementation.***

* Programming language: Java

***ii. Operating system to test the project.***

* Operating system: Mac OS/Windows/Linux

**[c] *Test cases.***

* Tested the accuracy and speed of communication working using IPC methods.
* Tested for large input.

**Experimental Results**

**[a] *Any results or output in graphs or tables or figures that shows results of   
improvements/implementations.***

Pipes: Below is the working model of Pipes

Text

Description automatically generated

Shared Memory:

Text

Description automatically generated

Text

Description automatically generated

Message Passing:

Text

Description automatically generated

Text

Description automatically generated

**[b] *Interpretation of the results.***

* Pipes have built-in synchronization, so when something interesting happens, your reads and writes will cause the program to freeze and unfreeze. With shared memory, it is simpler to operate asynchronously and merely periodically check for new data, but the cost is significantly more complicated code.
* With pipes, you can have an input stream that feeds into one operation and an output stream that leaves it. The second process only needs to wait for something to appear while it is idle. The drawback is that a pipe is required to connect each pair of processes.
* To notify the second process that there is data in shared memory, some sort of mechanism must be used. Events or polls can do this. Shared memory on Windows is different from shared memory on Unix because it cannot be accessed directly. Before using it or unlocking it, it must be locked. Other processes won't see your modifications if it is locked.

**Conclusion**

**[a] *What have been accomplished and what is still left to be done?***

We have achieved the implementation of Inter Process communication through shared memory, message passing, and pipes. We were able to facilitate exchange of information and co-operation between processes. We even pulled off half-duplex mechanism and full-duplex mechanism in the event of communication among processes. The results show the need for IPC for the effective working of Operating Systems. From the results, we could interpret that IPC is way effective and faster in shared memory. The scope for IPC would always remain high and there’s need for a robust & swift IPC mechanism to complement the rapidly improving Operating systems. A good ecosystem is always beneficial and communication within the ecosystem needs to be improved and communication between different ecosystems is also anticipated. These possibilities can come into play only with a faster and effective mechanism in IPC.

**References**

* <https://stackoverflow.com/questions/9701757/when-to-use-pipes-vs-when-to-use-shared-memory#:~:text=With%20pipes%20the%20synchronization%20is,of%20much%20more%20complex%20code>.
* <https://forums.codeguru.com/showthread.php?197005-Pipes-Vs-shared-memory>