Project 6: Inter-process communication (IPC) in pi-OS

ENEE 447: Operating Systems — Spring 2017
Assigned Date: Thursday, May 4th.
Due Date: **Sunday, May 14th. @11:59 p.m.**

1. Abstract

In project 5, we learned about process and thread, and implemented system calls to support multi-thread/process in pi-OS.

In your last project, you will learn and implement mechanism for processes to communicate with each other via inter-process communication (IPC). You will implement IPC with message passing approach in this project.

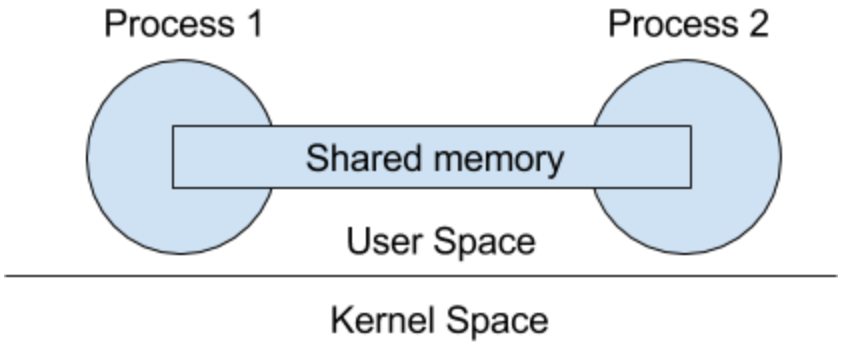
2. Introduction

From last discussion, we know that multiple processes helps to speed up computation of the program. We also know that different processes do not share address space, which means they are not able to share information via shared memory, like threads. However, real world applications usually required process communication between each other to share information/dat. Therefore, inter-process communication (IPC) is mechanism for processes to communicate with each other.

There are many of implementations of IPC and we are going to discuss the most popular two implementations for IPC: shared memory, and message passing. Let's briefly go over both of them.

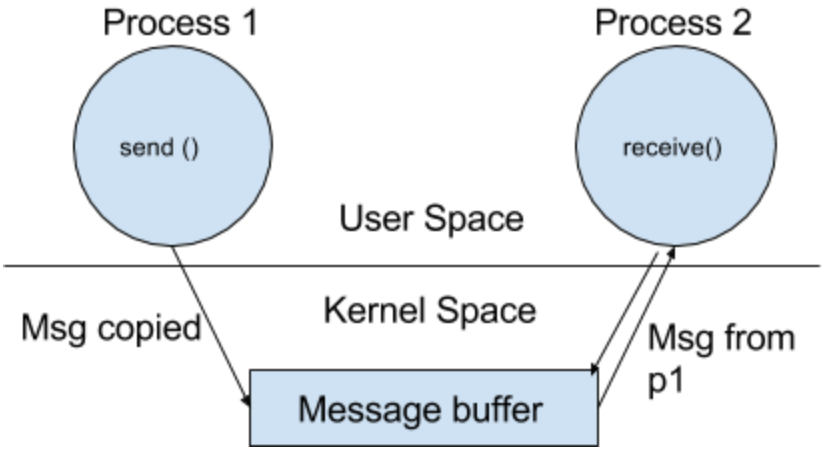
Shared Memory IPC

Shared memory IPC is achieved by allocating a shared space in memory for different processes. This shared area is used for sharing data between processes.



In shared memory IPC, the shared space is located in user space, which means operating system (OS) does not participate. Because of that, shared memory IPC is usually faster because no system calls are required. However, one downside of shared memory IPC is it tends to be more error prone and read/write access need to be carefully implemented to prevent concurrency issue.

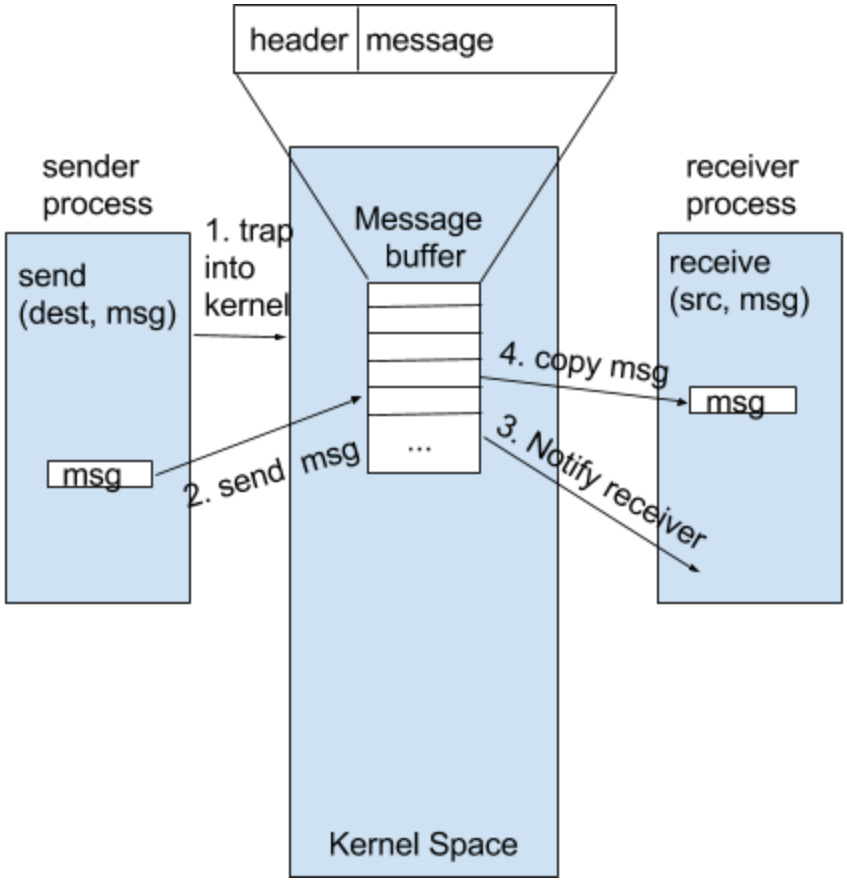
Message Passing IPC



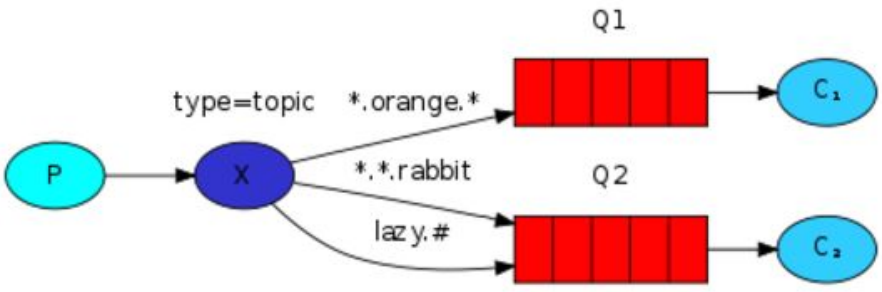
On the other hand, IPC via message passing do require OS's management because the message buffer is located in kernel space. User process should never be able to access kernel space directly. Instead, it should invoke a syscall to ask help from from operating system. In order for process 1 to send a message to process 2, system call *send()* is required. Upon calling *send()* syscall, process 1's message will be copied into an empty slot in message buffer. Similarly, in order for process 2 to receive the message sent by process 1, an explicit syscall *receive()* is also required.

Obviously, message passing IPC is more easily to implement because processes will not run into each other and no concurrence issue need to be concerned. However, the downside of this approach is its overhead in performance because system calls are expensive.

Message Passing (General model):



Message Passing (Project6 topics subscriber model)



Refer to the ***Interprocess Communication Design.pdf*** for implementation detail.

3. Your mission

To support message passing IPC in pi-OS, there are following steps:

- understand IPC model in project 6.
- new system calls supported:
    - *Send(topic, message*)*,
    - *Receive(topic, message*),*

4. Implementation guide:

Please refer to ***Interprocess Communication Design.pdf*** for implementation details.

5. Submission

No extension, any late submission (after **Sunday, May 14th. @11:59 p.m.**) will result in penalty. If you submit one day late, 10% will be deducted from your project score. If you submit two days late, 20% will be deducted, etc.

What is inside your zip submission:

- Report should includes 1. output screenshots, 2. brief explanation of the outputs, 3. answers to questions on Piazza (will be post soon) in **PDF format**.
- All Project files, make sure it is compliable, any compile error automatically 10% off.

Demo:

- There will be no demo for project 6. Instead, you are required to answer related questions in your report.

6. Honor Code

Copying code and idea is considering cheating and will be reported to Student Honor Council. The University of Maryland, College Park has a nationally recognized Code of Academic Integrity. This Code sets standards for academic integrity at Maryland for all undergraduate and graduate students. As a student you are responsible for upholding these standards for this course. It is very important for you to be aware of the consequences of cheating, fabrication, facilitation, and plagiarism. For more information on the Code of Academic Integrity or the Student Honor Council, please visit http://www.shc.umd.edu

7. Reference:

message passing vs shared memory
http://stackoverflow.com/questions/1853284/whats-the-difference-between-the-message-passing-and-shared-memory-concurrency