BÀI 16: GIAO TIẾP LIÊN TIẾN TRÌNH (IPC)

Inter-Process Communication

🎯 Mục tiêu bài học

1. **Vấn đề:** Tại sao Process A không thể đọc biến của Process B? (Virtual Memory Protection).
2. **Pipe:** Đường ống dẫn dữ liệu đơn giản.
3. **Message Queue:** Hàng đợi tin nhắn (POSIX vs System V).
4. **Shared Memory:** Chia sẻ vùng nhớ chung (Nhanh nhất).

1. Pipe (Đường ống)

Dùng để truyền dữ liệu một chiều giữa Cha và Con.

```
int fd[2];
pipe(fd);  // Tạo đường ống: fd[0] đọc, fd[1] ghi

if (fork() == 0) {
    // Con: Đóng đầu đọc, ghi vào đầu ghi
    close(fd[0]);
    write(fd[1], "Hello Parent", 12);
} else {
    // Cha: Đóng đầu ghi, đọc từ đầu đọc
    close(fd[1]);
    char buf[100];
    read(fd[0], buf, 100);
}
```

“ **Named Pipe (FIFO):** Pipe có tên file, dùng cho 2 process không liên quan. ”

2. Message Queue (Hàng đợi)

Giống hòm thư. Process A gửi thư vào, Process B lấy thư ra. Tin nhắn có cấu trúc.

- **Tạo Queue:** `mq_open()`
- **Gửi:** `mq_send()`
- **Nhận:** `mq_receive()`

> **Ưu điểm:** Bất đồng bộ (A gửi xong đi làm việc khác, B lúc nào rảnh thì đọc).

3. Shared Memory (Bộ nhớ chia sẻ)

Cách nhanh nhất để trao đổi lượng lớn dữ liệu (Ví dụ: Video Frame từ Camera process sang Display process).

- **Cơ chế:** Map một vùng RAM vật lý vào không gian địa chỉ của cả 2 process.
- **Lưu ý:** Phải dùng **Semaphore** hoặc **Mutex** để đồng bộ (tránh Race Condition).

🛠️ PHẦN THỰC HÀNH (LAB 16)

Chat 2 Process qua Named Pipe

Yêu cầu

1. Tạo 2 chương trình: `sender.c` và `receiver.c`.
2. Tạo một FIFO: `mkfifo my_chat_pipe`.
3. **Sender:** Mở Pipe, cho người dùng nhập tin nhắn từ bàn phím → Ghi vào Pipe.
4. **Receiver:** Mở Pipe, đọc liên tục → In ra màn hình.
5. Thử chạy trên 2 terminal khác nhau.

Q & A

Hẹn gặp lại ở Bài 17: Lập trình Mạng!