KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG MÁY TÍNH VÀ TT DỮ LIỆU

Hướng dẫn Lab 5.3 – Vùng nhớ chia sẻ

Với đường ống, việc liên lạc có 2 đặc trưng: đường liên lạc giữa 2 bên và Dữ liệu gửi theo thứ tự FIFO. Nếu có từ 3 tiến trình trở lên, hoặc dữ liệu cần truy cập không theo thứ tự, Vùng nhớ chia sẻ (Shared memory) là giải pháp phù hợp. Hơn nữa, vùng nhớ chia sẻ cho phép lưu trữ các cấu trúc dữ liệu tuỳ biến struct do người dùng định nghĩa.

Mục tiêu	Lý thuyết liên quan	Tài nguyên
Vùng nhớ chia sẻ POSIX	Ch3: Process Ch3.3.7.1 POSIX Shared Memory	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18 Sử dụng Windows 10
Bài toán Producer – Consumer		
Đọc thêm: SM trong System-V		

Yêu cầu sinh viên: Hiểu lý thuyết về liên lạc giữa các tiến trình bằng phương pháp vùng nhớ chia sẻ. Hiểu và thực thi các đoạn mã đã cung cấp. Áp dụng cho các bài toán liên lạc giữa các tiến trình. Phân tích và so sánh Vùng nhớ chia sẻ và các phương pháp liên lạc khác.

Đánh giá sinh viên: Hỏi đáp các vấn đề lý thuyết. Kỹ năng thực hành. Bài tập.

Yêu cầu nộp bài: các tập tin mã nguồn .c và tập tin khả thực thi .out của các Yêu cầu trong buổi thực hành và Bài tập cuối hướng dẫn trong thời gian cho phép của giảng viên.

Preferences

[1] Abraham Silberschatz, Peter B. Galvin, Greg Gagne, [2018], Operating System Concepts, 10th edition, John Wiley & Sons, New Jersey.

Programming Problems of Chapter 3.

[2] Greg Gagne , [2019], GitHub OS-BOOK OSC10e, Westminster College, United States

Access https://github.com/greggagne/osc10e in September 2019.

[3] Linux manual page, [2021], on *mmap(2)*

Access https://man7.org/linux/man-pages/man2/mmap.2.html

[4] [Linux manual page, [2021], on shm_open(3)

Access https://man7.org/linux/man-pages/man3/shm_open.3.html

Yêu cầu 1: Hoàn tất chương trình trong đó tiến trình con đọc vào 2 số nguyên từ đối số truyền, ghi vào SM, tiến trình cha thực hiện tính tổng và ghi lại vào SM. Tiến trình con đọc kết quả và xuất ra màn hình

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_5/task3_1.c***, biên dịch và thực thi.

```
>gcc -o task3_1.out task3_1.c -lrt
>./task3_1.out 1 2
1 + 2 = 3
```

- Dòng 16, 18 và 22 là các thư viện liên quan.

- Dòng 24 là kích thước vùng nhớ chia sẻ muốn tạo ra.

- Dòng 23 định nghĩa con trỏ * shm để truy cập vùng nhớ, shm_id là ID vùng nhớ.

- Dòng 28 là khoá, các tiến trình chỉ có thể truy cập vùng nhớ nếu cùng giá trị khoá (là một chuỗi định trước).

- Dòng 29 tạo ra bộ nhớ chia sẻ bằng cách tạo ra một segment mới (O_CREAT), có quyền đọc ghi (O_RDWR) cấp cho cả 3 nhóm owner / group / other. Lưu ý: là 0666 chứ không phải 666.

- Dòng 34 định ra kích thước SIZE cho vùng nhớ.

- Dòng 36 ánh xạ vùng nhớ bắt đầu từ địa chỉ cục bộ (tham số đầu tiên, ở đây là 0 nghĩa là đầu vùng nhớ đã tạo rao); có kích thước là tham số thứ hai (ở đây là SIZE); được dịch đi một đoạn offset là tham số cuối cùng (ở đây là 0), trả về con trỏ quản lý (ở đây là shm). ID của vùng nhớ truyền vào tham số thứ năm đã được tạo ra ở dòng 23.

  Tham số thứ ba: PROT_WRITE cho phép ghi; trong khi PROT_EXEC cho phép thực thi, PROT_READ chỉ cho phép đọc còn PROT_NONE từ chối mọi truy cập.

  Tham số thứ tư: MAP_SHARED chia sẽ thông tin ánh xạ này cho mọi tiến trình, còn MAP_PRIVATE sẽ "copy-on-write" thông tin ánh xạ này dành riêng cho tiến trình gọi.

- Dòng 40, 42 và 48 là các thao tác trên bộ nhớ chia sẻ.

- Dòng 44 và 49 là lệnh gỡ (unlink) con trỏ điều khiển ra khỏi vùng nhớ chia sẻ của tiến trình con và cha, sau khi hoàn tất mọi thao tác.

- Dòng 34 là lệnh huỷ bỏ vùng nhớ chia sẻ, trước đó, tất cả tiến trình phải kết thúc con trỏ trỏ đến vùng nhớ này (unlink()).

```
16      #include <sys/shm.h>
18      #include <fcntl.h>
22      #include <sys/mman.h>
24      #define SIZE 256

27        int * shm, shm_id, k, pid;
28        const char * key = "sharedkey";
29        shm_id = shm_open(key, O_CREAT | O_RDWR, 0666))
34        ftruncate(shm_id, SIZE);
36        shm = mmap(0, SIZE, PROT_WRITE, MAP_SHARED, shm_id,
        0);

40          shm[0] = atoi(argv[1]);
42          shm[1] = atoi(argv[2]);
44          shm_unlink(key);
48          shm[2] = shm[1] + shm[0];
49          shm_unlink(key);
```

Yêu cầu 2: Thực hành bài toán Producer – Consumer bằng phương pháp Vùng nhớ chia sẻ

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/CHAPTER_3/Fig3_16_Producer_POSIX_SHM.c*** và ***Fig3_17_Consumer_POSIX_SHM.c***, biên dịch và thực thi trên 2 terminal riêng biệt.

a. Giải thích mã nguồn PARENT

- Dòng 22 định nghĩa kích thước vùng nhớ (4096 byte).

- Dòng 24 là khoá của vùng nhớ.

- Dòng 26, 27 định nghĩa 2 chuỗi kí tự.

- Dòng 16 là sự cập nhật biến value thực hiện bởi tiến trình con.

- Dòng 31 khai báo con trỏ quản lý vùng nhớ, do thao tác trên từng kí tự, nên char * được sử dụng.

- Dòng 39 ghi chuỗi kí tự thứ nhất vào vùng nhớ, dòng 40 dịch con trỏ ra sau cuối chuỗi rồi tiếp tục ghi chuỗi kí tự thứ hai bằng dòng 41.

```
22        const int SIZE = 4096;
24        const char * name = "OS";
26        const char * message_0 = "Hello";
27        const char * message_1 = "World!";
31        char * ptr;
39        sprintf(ptr, "%s", message_0);
40        ptr += strlen(message_0);
41        sprintf(ptr, "%s", message_1);
42        ptr += strlen(message_1);
```

b. Giải thích mã nguồn CHILD

** Đoạn code tại trang 134 sách [1] (hình 3.17) có lỗi, dòng ptr = (char *) mmap(0, SIZE, PROT_READ ~~| PROT_WRITE~~ , MAP_SHARED , fd, 0); cần loại bỏ đi thuộc tính PROT_WRITE.*

- Dòng 26 khai báo con trỏ thao tác với chuỗi kí tự.

- Dòng 30 ánh xạ vào vùng nhớ chia sẻ, chỉ đọc và gán cho ptr.

- Dòng 33 in ra màn hình nội dung vùng nhớ chia sẻ.

- Dòng 35 huỷ bỏ liên kết với vùng nhớ chia sẻ.

```
26      ptr = (char * )
30      mmap(0, SIZE, PROT_READ , MAP_SHARED, fd, 0);
        /* read from the shared memory object */
33      printf("%s", (char * ) ptr);
35      shm_unlink(name);
```

Bài tập lập trình.

1. Tiến trình cha chuyển đối số đầu tiên (argv [1]) là một số nguyên lớn hơn 3 cho tiến trình con thông qua vùng nhớ chia sẻ. Tiến trình con nhận, tính giá trị n! = 1 * 2 * … * n và ghi nó vào vùng nhớ chia sẻ. Tiến trình cha nhận và xuất dữ liệu ra màn hình.

```
>./baitap2A.out 4
4! = 24
```

2. Tiến trình cha đọc hai số nguyên và một thao tác +, -, *, / và chuyển tất cả cho tiến trình con. Tiến trình con tính toán kết quả và trả về cho tiến trình cha để in ra màn hình. Việc liên lạc sử dụng một vùng nhớ chia sẻ.

```
>./baitap3A.out 4 6 +
4 + 6 = 10
```

3. Xây dựng 2 tiến trình Producer và Consumer, trong đó Producer gửi đi từng item là một câu (kích thước tối đa 40 kí tự), và Consumer nhận rồi in ra màn hình. Kích thước Buffer sử dụng là 10 (số câu tối đa chứa tạm là 10), Buffer được hiện thực bằng một vùng nhớ chia sẻ.