Họ và tên: Bùi Lê Nhật Tri

MSSV: 23521634

Lớp: IT00007.P11.1

BÁO CÁO LAB 3

Câu 1: Thực hiện Ví dụ 3-1, Ví dụ 3-2, Ví dụ 3-3, Ví dụ 3-4 giải thích đoạn code và kết quả nhận được?

Ví dụ 3-1:

Kết quả:

```
nhattri@nhattri-VirtualBox:~$ ./test_fork ThamSo1 ThamSo2 ThamSo3
PARENTS | PID = 5706 | PPID = 3287
PARENTS | There are 3 arguments
CHILDREN | PID = 5707 | PPID = 5706
CHILDREN | List of arguments:
ThamSo1
ThamSo2
ThamSo3
nhattri@nhattri-VirtualBox:~$
```

Giải thích:

1. Tạo tiến trình con:

- **fork():** Tạo một bản sao giống hệt tiến trình hiện tại (gọi là tiến trình con).
- **pid:** Lưu trữ ID của tiến trình con.

2. Tiến trình cha:

- **In thông tin:** In ra ID của chính nó và ID của tiến trình con.
- **Kiểm tra tham số:** Nếu có nhiều hơn 2 tham số truyền vào thì in ra số lượng.
- **Đợi con:** Chờ cho tiến trình con kết thúc.

3. Tiến trình con:

- **In thông tin:** In ra ID của chính nó và ID của tiến trình cha.
- **In tham số:** In ra danh sách các tham số được truyền vào.

Ví dụ 3-2:

Kết quả:

```
nhattri@nhattri-VirtualBox:~$ ./test_execl ThamSo1 ThamSo2 ThamSo3
PARENTS | PID = 6330 | PPID = 3287
PARENTS | There are 3 arguments
Implementing: ./count.sh28
PPID of count.sh:
nhattri     6331    6330  0 14:47 pts/0    00:00:00 /bin/bash ./count.sh 10
nhattri     6333    6331  0 14:47 pts/0    00:00:00 grep count.sh
```

Giải thích:

Thay thế tiến trình con (tiến trình con):

- **execl():** Hàm này thay thế hoàn toàn tiến trình hiện tại bằng một tiến trình mới được khởi chạy từ một file thực thi.
- **./count.sh:** Đường dẫn đến file thực thi (script shell) cần chạy.
- **"./count.sh" và "10":** Các tham số được truyền vào script shell.
- **NULL:** Kết thúc danh sách tham số.
- **perror("execl"):** In ra thông báo lỗi nếu execl() thất bại.

Ví dụ 3-3:

Kết quả:

```
nhattri@nhattri-VirtualBox:~$ ./test_system ThamSo1 ThamSo2 ThamSo3
PARENTS | PID = 6702 | PPID = 3287
PARENTS | There are 3 arguments
Implementing: ./count.sh28
PPID of count.sh:
nhattri     6703    6702  0 14:49 pts/0    00:00:00 /bin/bash ./count.sh 10
nhattri     6705    6703  0 14:49 pts/0    00:00:00 grep count.sh
nhattri@nhattri-VirtualBox:~$
```

Giải thích:

 Thông tin tiến trình cha: In ra PID và PPID của tiến trình cha.
 Kiểm tra tham số: Kiểm tra số lượng tham số được truyền vào chương trình test_system.c (có nhiều hơn 2 không) và in ra thông báo nếu có.
 Chạy script shell:

- system("./count.sh 10"): Sử dụng hàm system() để chạy script shell ./count.sh với tham số "10".
    - Hàm system() sẽ:
        - Tạo một tiến trình con mới.
        - Thực thi script shell ./count.sh trong tiến trình con.
        - Đợi cho tiến trình con kết thúc.
        - Trả về trạng thái thoát của script shell.

Ví dụ:3-4:

Kết quả:

```
nhattri@nhattri-VirtualBox:~$ ./test_shm_A
Waiting Process B update shared memory
Waiting Process B update shared memory
Waiting Process B update shared memory
Waiting Process B update shared memory
Waiting Process B update shared memory
Waiting Process B update shared memory
Memory updated: Hello Process A
nhattri@nhattri-VirtualBox:~$ ./test_shm_B
Read shared memory: Hello Process B
Shared memory updated: Hello Process A
nhattri@nhattri-VirtualBox:~$
```

Giải thích:

Tiến trình A tạo một vùng nhớ chia sẻ, viết dữ liệu vào đó, và chờ cho đến khi tiến trình B cập nhật dữ liệu. Sau đó, tiến trình A đọc và in ra nội dung mới của vùng nhớ chia sẻ.

Câu 2: Viết chương trình time.c thực hiện đo thời gian thực thi của một lệnh

shell. Chương trình sẽ được chạy với cú pháp "./time " với là lệnh shell muốn

đo thời gian thực thi.

Kết quả:

```
nhattri@nhattri-VirtualBox:~$ gcc time.c -o time
nhattri@nhattri-VirtualBox:~$ ./time "ls"
Desktop    Downloads  Music  Pictures  snap       test2.sh  time.c
Documents  LAB2       Myweb  Public    Templates  time      Videos
Thời gian thực thi: 0.00248 giây
```

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

#include <sys/wait.h>
#include <unistd.h>
```

```c
#include <sys/time.h>
int main(int argc, char *argv[])
{
    if (argc < 2)
    {
        fprintf(stderr, "Usage: %s <command>\n", argv[0]);
        exit(1);
    }
    struct timeval start, end;
    pid_t pid;
    // Lấy thời gian bắt đầu
    gettimeofday(&start, NULL);
    pid = fork();
    if (pid < 0)
    {
        perror("fork");
        exit(1);
    }
    else if (pid == 0)
    {
        // Tiến trình con thực thi lệnh shell
        execl("/bin/sh", "sh", "-c", argv[1], (char *)NULL);
        perror("execl");
        exit(1);
    }
    else
    {
        // Tiến trình cha đợi tiến trình con hoàn thành
        wait(NULL);
        // Lấy thời gian kết thúc
        gettimeofday(&end, NULL);
        // Tính toán thời gian thực thi
        double elapsed_time = (end.tv_sec - start.tv_sec) +
                              (end.tv_usec - start.tv_usec) / 1000000.0;
        printf("Thời gian thực thi: %.5f giây\n", elapsed_time);
    }
    return 0;
}
```

Câu 3: Viết một chương trình làm bốn công việc sau theo thứ tự:

- In ra dòng chữ: “Welcome to IT007, I am <your_Student_ID>!”

• Thực thi file script count.sh với số lần đếm là 120

• Trước khi count.sh đếm đến 120, bấm CTRL+C để dừng tiến trình này

• Khi người dùng nhấn CTRL+C thì in ra dòng chữ: “count.sh has stopped”

Kết quả:

```
nhattri@nhattri-VirtualBox:~$ chmod +x count.sh
nhattri@nhattri-VirtualBox:~$ ./test
Welcome to IT007, I am 21521652!
Count: 1
Count: 2
Count: 3
Count: 4
^C
count.sh has stopped
```

Code:

-File test.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
pid_t child_pid;
// Hàm xử lý tín hiệu SIGINT (CTRL+C)
void handle_sigint(int sig)
{
   if (child_pid > 0)
   {
      kill(child_pid, SIGKILL);
      printf("\ncount.sh has stopped\n");
   }
   exit(0);
}
int main()
{
   printf("Welcome to IT007, I am 21521652!\n");
   signal(SIGINT, handle_sigint);
   child_pid = fork();
   if (child_pid == 0)
```

```
    {
        execl("./count.sh", "count.sh", NULL);
        perror("execl");
        exit(1);
    }
    else if (child_pid > 0)
    {
        wait(NULL);
    }
    else
    {
        perror("fork");
        exit(1);
    }
    return 0;
}
```

-File count.sh:

```
#!/bin/sh
# count.sh

for i in $(seq 1 120); do
    echo "Count: $i"
    sleep 1
done
```

Câu 4: Viết chương trình mô phỏng bài toán Producer - Consumer như sau:

• Sử dụng kỹ thuật shared-memory để tạo một bounded-buffer có độ lớn là

10 bytes.

• Tiến trình cha đóng vai trò là Producer, tạo một số ngẫu nhiên trong

khoảng [10, 20] và ghi dữ liệu vào buffer

• Tiến trình con đóng vai trò là Consumer đọc dữ liệu từ buffer, in ra màn

hình và tính tổng

• Khi tổng lớn hơn 100 thì cả 2 dừng lại|

Kết quả:

```
nhattri@nhattri-VirtualBox:~$ gcc test.c -o test
nhattri@nhattri-VirtualBox:~$ ./test
Producer produced: 14
Consumer consumed: 14, Current Sum: 14
Producer produced: 18
Consumer consumed: 18, Current Sum: 32
Producer produced: 11
Consumer consumed: 11, Current Sum: 43
Producer produced: 19
Consumer consumed: 19, Current Sum: 62
Producer produced: 13
Consumer consumed: 13, Current Sum: 75
Producer produced: 18
Consumer consumed: 18, Current Sum: 93
Producer produced: 20
Consumer consumed: 20, Current Sum: 113
Producer produced: 10
Producer and Consumer have stopped.
```

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>

#define BUFFER_SIZE 10
typedef struct
{
    int buffer[BUFFER_SIZE];
    int in;
    int out;
    int sum;
} SharedMemory;
int main()
{
    int shm_id;
    SharedMemory *shared_mem;
    // Tạo vùng nhớ chia sẻ
    shm_id = shmget(IPC_PRIVATE, sizeof(SharedMemory), IPC_CREAT | 0666);
    if (shm_id < 0)
    {
        perror("shmget failed");
        exit(1);
    }
```

```
shared_mem = (SharedMemory *)shmat(shm_id, NULL, 0);
if (shared_mem == (SharedMemory *)-1)
{
    perror("shmat failed");
    exit(1);
}
// Khởi tạo buffer và các chỉ số
shared_mem->in = 0;
shared_mem->out = 0;
shared_mem->sum = 0;
pid_t pid = fork();
if (pid < 0)
{
    perror("fork failed");
    exit(1);
}
else if (pid == 0)
{ // Tiến trình con - Consumer
    while (1)
    {
        // Kiểm tra xem có dữ liệu trong buffer không
        if (shared_mem->in != shared_mem->out)
        {
            int item = shared_mem->buffer[shared_mem->out];
            shared_mem->out = (shared_mem->out + 1) % BUFFER_SIZE;
            // Tính tổng và in ra màn hình
            shared_mem->sum += item;
            printf("Consumer consumed: %d, Current Sum: %d\n", item, shared_mem->sum);
            // Dừng nếu tổng vượt quá 100
            if (shared_mem->sum > 100)
            {
                break;
            }
        }
        usleep(100000); // Đợi để tránh busy-waiting
    }
    shmdt(shared_mem);
    exit(0);
}
else
{ // Tiến trình cha - Producer
    srand(time(NULL));
    while (1)
    {
        // Tạo số ngẫu nhiên từ 10 đến 20
        int item = rand() % 11 + 10;
        // Kiểm tra nếu buffer không đầy
```

```
            if ((shared_mem->in + 1) % BUFFER_SIZE != shared_mem->out)
            {
                shared_mem->buffer[shared_mem->in] = item;
                shared_mem->in = (shared_mem->in + 1) % BUFFER_SIZE;
                printf("Producer produced: %d\n", item);
            }
            // Kiểm tra nếu tổng đã vượt quá 100 để dừng
            if (shared_mem->sum > 100)
            {
                break;
            }
            usleep(200000); // Đợi để tránh busy-waiting
        }
        // Đợi tiến trình con hoàn thành
        wait(NULL);
        // Hủy vùng nhớ chia sẻ
        shmdt(shared_mem);
        shmctl(shm_id, IPC_RMID, NULL);
        printf("Producer and Consumer have stopped.\n");
    }
    return 0;
}
```