BÀI BÁO CÁO THỰC HÀNH 3 – HỆ ĐIỀU HÀNH

CLASS : IT007.O26.1

TEACHER : MR. ĐOÀN DUY

Date: 20/04/2024

Reported by: Trần Ngọc Ánh | 22520077 | 22520077@gm.uit.edu.vn

SUMMARY

A. Bài tập thực hành

1. Thực hiện ví dụ 3-1, ví dụ 3-2, ví dụ 3-3, ví dụ 3-4 giải thích code và kết quả nhận được?

- Ví dụ 3-1:

File test_fork.c

```
/*#######################################
# University of Information Technology
# IT007 Operating System
#
# <Tran Ngoc Anh>, <22520077>
# File: test_fork.c
#
#######################################*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>  // sử dụng hàm fork()
#include <sys/wait.h>
#include <sys/types.h>
```

```c
int main(int argc, char *argv[])  // argc là số lượng đối số được truyền vào
{
    // Tạo một tiến trình con bằng cách sử dụng hàm fork()
    __pid_t pid;
    pid = fork();

    // Đây là phần mã của tiến trình cha
    if (pid > 0)
        {
        // In ra pid và ppid của tiến trình hiện tại
        printf("parents | pid = %ld | ppid\n", (long)getpid(),
(long)getppif());

        // Nếu có nhiều hơn 2 đối số in ra số lượng đối số
        if (argc > 2)
            printf("PARENTS | There are %d arguments\n", argc - 1);

        // Tiến trình cha đợi tiến trình con kết thúc
        wait(NULL);
    }

    // Thông tin về tiến trình con và các đối số được in ra
    if (pid == 0)
    {
        printf("CHILDREN | PID = %ld | PPID = %ld\n", (long)getpid(),
(long)getppid());
        printf("CHILDEN | List of arguments: \n");

        // In ra danh sách các đối số được truyền vào chương trình
        for (int i = 1; i < argc; i++)
        {
            printf("%s\n", argv[i]);
        }
    }
    exit(0);
}
```

1. Đây là thông tin về tiến trình cha (parent process). PID = 694 là ID của tiến trình cha và PPID = 177 là ID của tiến trình ông nội (grandparent process).
2. Cho biết rằng bạn đang chạy chương trình với 3 đối số đầu vào: ThamSo1, ThamSo2, ThamSo3.

3. Thông tin về tiến trình con (child process). PID = 695 là ID của tiến trình con và PPID = 694 là ID của tiến trình cha.
4. Đây là danh sách các đối số đầu vào mà bạn đã truyền vào chương trình. Trong trường hợp này, các đối số là ThamSo1, ThamSo2, ThamSo3.

```
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc -o test test_fork.c
ngocanh-22520077@TNANH-D53R8MB9:~$ ./test ThamSo1 ThamSo2 ThamSo3
1. PARENTS | PID = 694 | PPID = 177
2. PARENTS | There are 3 arguments
3. CHILDREN | PID = 695 | PPID = 694
4. CHILDREN | List of arguments:
ThamSo1
ThamSo2
ThamSo3
ngocanh-22520077@TNANH-D53R8MB9:~$
```

- Ví dụ 3-2:

File count.sh

```
#!/bin/bash
# Đây là shebang, chỉ định script này sẽ được thực thi bằng /bin/bash

echo "Implementing: $0"
# In ra tên của script hiện tại

echo "PPID of count.sh: "
# In ra chuỗi PPID of count.sh

ps -ef | grep count.sh
# Lệnh ‘ps -ef’ sẽ liệt kê tất cả các tiến trình đang chạy
# Kết quả sau đó được chuyển đến lệnh ‘grep count.sh’
# Lọc ea những dòng có chứa chuôi ‘count.sh’

i=1
# Khởi tạo biến i với giá trị là 1

while [ $i -le $1 ]
# Vòng lặng while sẽ chạy miễn là giá trị của i nhỏ hơn hoặc bằng tham số
đầu tiên ($1) được truyền vào script
```

```
do
    echo $i >> count.txt
    # In ra giá trị của i vào file count.txt. '>>' có nghĩa là nối vào
cuối file nếu file đã tồn tại.

    i=$((i + 1))
    # Tăng giá trị i lên 1

    sleep 1
    # Dừng script trong 1 giây

done
# Kết thúc vòng lặp while

exit 0
# Kết thúc script với mã thoát là 0, biểu thị rằng script đã hoàn thành
mà không có lỗi xảy ra.
```

File test_excel.c

```
/*########################################
# University of Information Technology
# IT007 Operating System
#
# <Tran Ngoc Anh>, <22520077>
# File: test_execl.c
#
########################################*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

int main(int argc, char* argv[])
{
    // Tạo một tiến trình con bằng cách sử dụng hàm fork()
    __pid_t pid;
    pid = fork();

    // Đây là phần mã của tiến trình cha
    if (pid > 0)
    {
        // In ra PID và PPID của tiến trình cha
```

```
        printf("PARENTS | PID = %ld | PPID = %ld\n", (long)getpid(),
(long)getppid());

        // Nếu có nhiều hơn 2 đối số, in ra số lượng đối số
        if (argc > 2)
        printf("PARENTS | There are %d arguments\n", argc - 1);

        // Tiến trình cha đợi tiến trình con kết thúc
        wait(NULL);
    }

    // Đây là phần mã của tiến trình con
    if (pid == 0)
    {
         // Tiến trình con thực thi script ‘count.sh’ với đối số là ‘10’
         execl("./count.sh", "./count.sh", "10", NULL);

         // In ra PID và PPID của tiến trình con
         printf("CHILDREN | PID = %ld | PPID = %ld\n", (long)getpid(),
(long)getppid());
         printf("CHILDREN | List of arguments: \n");

         // In ra danh sách các đối số được truyền vào chương trình
         for (int i = 1; i < argc; i++)
         {
             printf("%s\n", argv[i]);
         }
    }

    exit(0);
    // Kết thúc chương trình với mã thoát là 0, biểu thị rằng chương
trình đã hoàn thành mà không có lỗi xảy ra
}
```

1. Output từ tiến trình cha trong chương trình test_excel.c. Nó in ra PID (process ID) của tiến trình cha là 6814 và PPID (parent process ID) của nó là 1923.
2. Output từ tiến trình cha, nó in ra số lượng đối số được truyền vào chương trình test_excel.c (không tính tên chương trình). Trong trường hợp này, bạn đã truyền vào 3 đối số: ThamSo1, ThamSo2, ThamSo3.
3. Output từ script bash ‘count.sh, nó in ra tên của script hiện tại.

4. Tiếp tục là output từ script bash ‘count.sh’, nó in ra chuỗi “PPID of count.sh: ”.
5. Đây là output từ lệnh ps -ef | grep count.sh trong script bash ‘count.sh’. Nó in ra thông tin về tiến trình đang chạy script ‘count.sh’. Trong đó, PID của tiến trình này là 6815 và PPID của nó (là tiến trình cha đã tạo ra nó) là 6814.
6. Đây cũng là output từ lệnh ps -ef | grep count.sh, nhưng nó liên quan đến tiến trình grep mà lệnh này đã tạo ra để lọc kết quả. PID của tiến trình grep này là 6817 và PPID của nó (tiến trình cha đã tạo ra nó) là 6815.

```
ngocanh-22520077@TNANH-D53R8MB9:~$ chmod +x count.sh
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc -o test test_excel.c
ngocanh-22520077@TNANH-D53R8MB9:~$ ./test ThamSo1 ThamSo2 ThamSo3
1.PARENTS | PID = 6814 | PPID = 6496
2.PARENTS | There are 3 arguments
3.Implementing: ./count.sh
4.PPID of count.sh:
5.ngocanh+  6815  6814  0 14:46 pts/3    00:00:00 /bin/bash ./count.sh 10
6.ngocanh+  6817  6815  0 14:46 pts/3    00:00:00 grep count.sh
ngocanh-22520077@TNANH-D53R8MB9:~$
```

- Ví dụ 3-3:

File test_system.c:

```c
/*########################################
# University of Information Technology
# IT007 Operating System
#
# <Tran Ngoc Anh>, <22520077>
# File: test_system.c
#
########################################*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>
int main(int argc, char* argv[])
{
    // In ra ID của tiến trình hiện tại và ID của tiến trình cha (nếu có)
```

```
    printf("PARENTS | PID = %ld | PPID = %ld\n", (long)getpid(),
(long)getppid());

    // Nếu có nhiều hơn 2 tham số đầu vào, in ra số lượng tham số
    if (argc > 2)
    {
        printf("PARENTS | There are %d arguments\n", argc - 1);
    }

    // Thực thi script shell count.sh với tham số 10 trong tiến trình con
    system("./count.sh 10");

    // In ra danh sách các tham số đầu vào
    printf("PARENTS | List of arguments: \n");
    for (int i = 1; i < argc; i++)
    {
        printf("%s\n", argv[i]);
    }

    // Kết thúc chương trình với mã lỗi 0 (tức không có lỗi xảy ra)
    exit(0);
}
```

1. Đây là thông tin về tiến trình cha (parent process). PID = 10376 là ID của tiến trình cha và PPID = 10070 là ID của tiến trình ông nội (grandparent process).
2. Cho biết rằng bạn đã chạy chương trình với 3 đối số đầu vào là ThamSo1, ThamSo2, ThamSo3.
3. Cho biết rằng chương trình đang thực thi script shell count.sh.
4. Thông tin về các tiến trình liên quan đến việc thực thi script shell count.sh. Cụ thể, sh -c ./count.sh 10 là tiến trình shell được tạo ra bởi hàm system() để thực thi script và /bin/bash ./count.sh 10 là tiến trình của script shell count.sh đang chạy.
5. Danh sách các đối số đầu vào mà bạn đã truyền vào chương trình.

```
ngocanh-22520077@TNANH-D53R8MB9:~$ chmod +x count.sh
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc -o test test_system.c
ngocanh-22520077@TNANH-D53R8MB9:~$ ./test ThamSo1 ThamSo2 ThamSo3
1. PARENTS | PID = 10376 | PPID = 10070
2. PARENTS | There are 3 arguments
3. Implementing: ./count.sh
4. PPID of count.sh:
   ngocanh+ 10377 10376  0 15:23 pts/5    00:00:00 sh -c ./count.sh 10
   ngocanh+ 10378 10377  0 15:23 pts/5    00:00:00 /bin/bash ./count.sh 10
   ngocanh+ 10380 10378  0 15:23 pts/5    00:00:00 grep count.sh
5. PARENTS | List of arguments:
   ThamSo1
   ThamSo2
   ThamSo3
ngocanh-22520077@TNANH-D53R8MB9:~$
```

- Ví dụ 3-4:

a) Process A (file test_shm_A.c):

```c
/*########################################
# University of Information Technology
# IT007 Operating System
#
# <Tran Ngoc Anh>, <22520077>
# File: test_shm_A.c
#
########################################*/

#include <stdio.h>      // Thư viện chuẩn C cho I/O
#include <stdlib.h>     // Thư viện chuẩn C cho các hàm như malloc, free, exit,…
#include <string.h>     // Thư viện chuẩn C cho các hàm xử lý chuỗi
#include <fcntl.h>      // Thư viện POSIX cho file control
#include <sys/shm.h>    // Thư viện POSIX cho shared memory
#include <sys/stat.h>   // Thư viện POSIX cho các hàm xử lý file status
#include <unistd.h>     // Thư viện POSIX cho các hàm như close, write, read,…

#include <sys/mman.h>   // Thư viện POSIX cho memory management

int main()
{
    // Kích thước (theo byte) của đối tượng bộ nhớ chia sẻ
    const int SIZE = 4096;

    // Tên của đối tượng bộ nhớ chia sẻ
    const char *name = "OS";
```

```c
    // File descriptor cho bộ nhớ chia sẻ
    int fd;

    // Con trỏ đến đối tượng bộ nhớ chia sẻ
    char *ptr;

    // Tạo đối tượng bộ nhớ chia sẻ
    fd = shm_open(name, O_CREAT | O_RDWR,0666);

    // Cấu hình kích thước của đối tượng bộ nhớ chia sẻ
    ftruncate(fd, SIZE);

    // Ánh xạ bộ nhớ của đối tượng bộ nhớ chia sẻ
    ptr = mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

    // Ghi vào đối tượng bộ nhớ chia sẻ
    strcpy(ptr, "Hello Process B");

    // Chờ cho đến khi process B cập nhật phân đoạn bộ nhớ chia sẻ
    while (strncmp(ptr, "Hello Process B", 15) == 0)
    {
        printf("Waiting Process B update shared memory\n");
        sleep(1);
    }
    printf("Memory updated: %s\n", (char *)ptr);

    // Hủy ánh xạ phân đoạn bộ nhớ chia sẻ và đóng file descriptor
    munmap(ptr, SIZE);
    close(fd);
    return 0;
}
```

Chương trình A sẽ tạo ra một đối tượng bộ nhớ chia sẻ và ghi chuỗi “Hello Process B” vào đó, sau đó liên tục in ra thông báo “Waiting Process B updated shared memory” cho đến khi chương trình B cập nhật bộ nhớ chia sẻ.

```
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc -o testA test_shm_A.c
ngocanh-22520077@TNANH-D53R8MB9:~$ ./testA
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
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
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
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
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
Waiting Process B update shared memory
Memory updated: Hello Process A
ngocanh-22520077@TNANH-D53R8MB9:~$
```

b) Process B (file test_shm_B.c):

```
/*########################################
# University of Information Technology
# IT007 Operating System
#
# <Tran Ngoc Anh>, <22520077>
# File: test_shm_B.c
```

```
#
#####################################*/

#include <stdio.h>      // Thư viện chuẩn C cho I/O
#include <stdlib.h>     // Thư viện chuẩn C cho các hàm như malloc, free, exit,…
#include <string.h>     // Thư viện chuẩn C cho các hàm xử lý chuỗi
#include <fcntl.h>      // Thư viện POSIX cho file control
#include <sys/shm.h>    // Thư viện POSIX cho shared memory
#include <sys/stat.h>   // Thư viện POSIX cho các hàm xử lý file status
#include <unistd.h>     // Thư viện POSIX cho các hàm như close, write, read,…

#include <sys/mman.h>   // Thư viện POSIX cho memory management

int main()
{
      // Kích thước (theo byte) của đối tượng bộ nhớ chia sẻ
      const int SIZE = 4096;

      // Tên của đối tượng bộ nhớ chia sẻ
      const char *name = "OS";

      // File descriptor cho bộ nhớ chia sẻ
      int fd;

      // Con trỏ đến đối tượng bộ nhớ chia sẻ
      char *ptr;

      // Mở đối tượng bộ nhớ chia sẻ đã tồn tại
      fd = shm_open(name, O_RDWR,0666);

      // Ánh xạ bộ nhớ của đối tượng bộ nhớ chia sẻ
      ptr = mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

      // Đọc từ đối tượng bộ nhớ chia sẻ
      printf("Read shared memory: ");
      printf("%s\n", (char *)ptr);

      // Cập nhật đối tượng bộ nhớ chia sẻ
     strcpy(ptr, "Hello Process A");
     printf("Shared memory updated: %s\n", ptr);
     sleep(5);

      // Hủy ánh xạ phân đoạn bộ nhớ chia sẻ và đóng file descriptor
      munmap(ptr, SIZE);
      close(fd);

      // Gỡ bỏ đối tượng bộ nhớ chia sẻ
```

```
    shm_unlink(name);
    return 0;
}
```

Chương trình B khi chạy sẽ đọc chuỗi từ bộ nhớ chia sẻ "Hello Process B", sau đó in ra chuỗi đã đọc và cập nhật bộ nhớ chia sẻ với chuỗi "Hello Process A" và in ra chuỗi mới.

```
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc -o testB test_shm_B.c
ngocanh-22520077@TNANH-D53R8MB9:~$ ./testB
Read shared memory: Hello Process B
Shared memory updated: Hello Process A
ngocanh-22520077@TNANH-D53R8MB9:~$
```

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 <command>" với <command> là lệnh shell muốn đo thời gian thực thi.

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

int main(int argc, char *argv[])
{
    // Kiểm tra xem có đúng một đối số được truyền vào không
    if (argc != 2)
    {
        printf("Cách sử dụng: ./time <lệnh>\n");
        return 1;
    }
    struct timeval start, end;

    // Lấy thời gian hiện tại và lưu vào biến start
    gettimeofday(&start, NULL);

    // Tạo một tiến trình con
    __pid_t pid = fork();

    // Nếu đây là tiến trình con
    if (pid == 0)
    {
        // Thực thi lệnh được truyền vào từ dòng lệnh
        if (execl("/bin/sh", "/bin/sh", "-c", argv[1], NULL) == -1)
        {
            printf("Lỗi: không thể thực thi lệnh\n");
```

```
            return 1;
        }
    }

    // Nếu đây là tiến trình cha
    else if (pid > 0)
    {
        // Chờ tiến trình con kết thúc
        wait(NULL);

        // Lấy thời gian hiện tại và lưu vào biến end
        gettimeofday(&end, NULL);

        // Tính thời gian thực thi bằng cách lấy thời gian kết thúc trừ thời
gian bắt đầu
        double ans = (end.tv_sec - start.tv_sec) + ((end.tv_usec -
start.tv_usec)/1000000.0);

        // In ra thời gian thực thi
        printf("Thời gian thực thi: %.5f giây\n", ans);
    }
    // Nếu có lỗi xảy ra khi tạo tiến trình con
    else
    {
        printf("Lỗi: không thể tạo tiến trình con\n");
        return 1;
    }
    return 0;
}
```

```
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc time.c -o time
ngocanh-22520077@TNANH-D53R8MB9:~$ ./time ls
 JPG                 hello.c                 testB
 Myweb               if_control.sh           test_excel.c
 PNG                 lab2_bai2.sh            test_fork.c
'Tran Ngoc Anh'      lab2_bai3.sh            test_permission.txt
 case1.sh            lab2_bai4.sh            test_shm_A.c
 case2.sh            lab2_btot1a.sh          test_shm_B.c
 case3.sh            lab2_createFolder.sh    test_system.c
 count.sh            monhoc.txt              time
 count.txt           move_and_count_png.sh   time.c
 dkhp.sh             move_countPNG.sh        try_variables.sh
 elif_control.sh     move_jpg.sh             until_user.sh
 elif_control2.sh    password.sh             variables.sh
 for_loop.sh         test                    while_for.sh
 for_loop2.sh        testA
Thời gian thực thi: 0.00158 giây
ngocanh-22520077@TNANH-D53R8MB9:~$
```

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>!”

```c
#include <stdio.h>
int main()
{
    char studentID[] = "22520077";
    printf("Welcome to IT007, I am %s!\n", studentID);
    return 0;
}
```

```
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc -o bai3 lab3_bai3.c
ngocanh-22520077@TNANH-D53R8MB9:~$ ./bai3
Welcome to IT007, I am 22520077!
ngocanh-22520077@TNANH-D53R8MB9:~$
```

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

```bash
#!/bin/bash

echo "Implementing: $0"
echo "PPID of count.sh: "
```

```
ps -ef | grep count.sh

i=1
while [ $i -le $1 ]
do
    echo $i >> count.txt
    i=$((i + 1))
    sleep 1
done
exit 0
```

```
ngocanh-22520077@TNANH-D53R8MB9:~$ chmod +x count.sh
ngocanh-22520077@TNANH-D53R8MB9:~$ ./count.sh 120
Implementing: ./count.sh
PPID of count.sh:
ngocanh+  5275  3589  0 17:38 pts/9    00:00:00 /bin/bash ./count.sh 120
ngocanh+  5277  5275  0 17:38 pts/9    00:00:00 grep count.sh
```

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

```
ngocanh-22520077@TNANH-D53R8MB9:~$ chmod +x count.sh
ngocanh-22520077@TNANH-D53R8MB9:~$ ./count.sh 120
Implementing: ./count.sh
PPID of count.sh:
ngocanh+  5275  3589  0 17:38 pts/9    00:00:00 /bin/bash ./count.sh 120
ngocanh+  5277  5275  0 17:38 pts/9    00:00:00 grep count.sh
^C
ngocanh-22520077@TNANH-D53R8MB9:~$
```

- File count.sh in ra “count. sh has stopped” khi có tín hiệu dừng

```
#!/bin/bash

echo "Implementing: $0"
echo "PPID of count.sh: "
ps -ef | grep count.sh

cleanup()
{
    echo "\n"
    echo "count.sh has stopped"
    exit 1
}
```

```
trap cleanup INT

i=1
while [ $i -le $1 ]
do
    echo $i >> count.txt
    i=$((i + 1))
    sleep 1
done
exit 0
```

```
ngocanh-22520077@TNANH-D53R8MB9:~$ chmod +x count.sh
ngocanh-22520077@TNANH-D53R8MB9:~$ ./count.sh 120
Implementing: ./count.sh
PPID of count.sh:
ngocanh+  8146  7540  0 17:50 pts/6    00:00:00 /bin/bash ./count.sh 120
ngocanh+  8148  8146  0 17:50 pts/6    00:00:00 grep count.sh
^C\n
count.sh has stopped
ngocanh-22520077@TNANH-D53R8MB9:~$
```

4. Viết chương trình mô phỏng bài toán Producer – Consumer như sau:

   - Sử dụng kỹ thuật shared-memory để tạo một boundedbuffer 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 lơn hơn 100 thì cả hai dừng lại.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/wait.h>

int main()
{
```

```
    const int BUFFER_SIZE = 10; // Định nghĩa kích thước buffer
    const char *name = "OS";    // Tên của shared-memory object

    // Tạo shared-memory object
    int shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);

    // Cấu hình kích thước của shared-memory object
    ftruncate(shm_fd, sizeof(int) * (BUFFER_SIZE + 1));

    // Mapping shared-memory object
    int *buffer = mmap(0, sizeof(int) * (BUFFER_SIZE + 1), PROT_READ |
PROT_WRITE, MAP_SHARED, shm_fd, 0);

    // Kiểm tra xem map có thành công không
    if (buffer == MAP_FAILED)
    {
        perror("mmap");
        return 1;
    }
    int *sum = buffer + BUFFER_SIZE;    // Lưu tổng vào cuối buffer
    *sum = 0;                           // Khởi tạo tổng bằng 0
    __pid_t pid = fork();               // Tạo một tiến trình con

    // Tiến trình con (consumer)
    if (pid == 0)
    {
        while (*sum <= 100)             // Chạy cho đến khi tổng lớn hơn 100
        {
            // Duyệt qua từng phần tử trong buffer
            for (int i = 0; i < BUFFER_SIZE; i++)
            {
                if (buffer[i] != 0)     // Nếu phần tử không rỗng
                {
                    printf("Consumer reads %d\n", buffer[i]); // In giá trị phần tử
                    *sum += buffer[i];  // Cộng giá trị phần tử vào tổng
                    buffer[i] = 0;      // Đặt lại giá trị phần tử thành 0
                }
            }
        }
        printf("Sum > 100, Consumer stops\n");  // In thông báo khi tổng > 100
    }
    else if (pid > 0)
    {
        // Tiến trình cha (Producer)
        while (*sum <= 100) // Chạy cho đến khi tổng lớn hơn 100
        {
            // Duyệt qua từng phần tử trong buffer
            for (int i = 0; i < BUFFER_SIZE; i++)
            {
```

```
                if (buffer[i] == 0)     // Nếu phần tử rỗng
                {
                    // Tạo một số ngẫu nhiên trong [10, 20] và gán vào phần tử
                    buffer[i] = rand() % 11 + 10;

                    // In ra giá trị đã ghi vào phần tử
                    printf("Producer writes %d\n", buffer[i]);
                }
            }
        }
        printf("Sum > 100, Producer stops\n");  // In thông báo khi tổng > 100
        wait(NULL);                             // Đợi tiến trình con kết thúc

        // Xóa shared-memory object
        shm_unlink(name);
    }
    else
    {
        // fork failed
        perror("fork");
        return 1;
    }
    return 0;
}
```

```
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc producer_consumer.c -o bai4
ngocanh-22520077@TNANH-D53R8MB9:~$ ./bai4
Producer writes 16
Producer writes 20
Producer writes 16
Producer writes 12
Producer writes 11
Producer writes 14
Producer writes 10
Producer writes 16
Producer writes 13
Producer writes 11
Consumer reads 16
Producer writes 18
Consumer reads 20
Consumer reads 16
Producer writes 17
Consumer reads 12
Producer writes 15
Consumer reads 11
Producer writes 13
Consumer reads 14
Producer writes 17
Consumer reads 10
Producer writes 14
Consumer reads 16
Producer writes 19
Consumer reads 13
Producer writes 20
Consumer reads 11
Producer writes 12
Sum > 100, Consumer stops
Producer writes 10
Sum > 100, Producer stops
ngocanh-22520077@TNANH-D53R8MB9:~$
```

B. Bài tập ôn tập

Phỏng đoán Collatz xem xét chuyện gì sẽ xảy ra nếu ta lấy một số nguyên dương bất kỳ và áp dụng theo thuật toán sau đây:

$$n = \begin{cases} n/2 & \text{nếu n là số chẵn} \\ 3 * n + 1 & \text{nếu n là số lẻ} \end{cases}$$

Phỏng đoán phát biểu rằng khi thuật toán này được áp dụng liên tục, tất cả số nguyên dương đều sẽ tiến đến 1. Ví dụ, với n = 35, ta sẽ có chuỗi kết quả như sau:

35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

Viết chương trình C sử dụng hàm fork() để tạo ra chuỗi này trong tiến trình con. Số bắt đầu sẽ được truyền từ dòng lệnh. Ví dụ lệnh thực thi ./collatz 8 sẽ chạy thuật toán trên n = 8 và chuỗi kết quả sẽ ra là 8, 4, 2, 1. Khi thực hiện, tiến trình cha và tiến trình con chia sẻ một buffer, sử dụng phương pháp bộ nhớ chia sẻ, hãy tính toán chuỗi trên tiến trình con, ghi kết quả vào buffer và dùng tiến trình cha để in kết quả ra màn hình. Lưu ý, hãy nhớ thực hiện các thao tác để kiểm tra input là số nguyên dương.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/mman.h>
#include <bits/mman-linux.h>

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        printf("Vui lòng nhập một số dương: ");
        return 1;
    }
    int n = atoi(argv[1]);
    if (n <= 0)
    {
        printf("Vui lòng nhập một số dương: ");
        return 1;
    }
    __pid_t pid;
    int *buffer;

    // Tạo bộ nhớ chia sẻ
    buffer = mmap(NULL, sizeof(int) * (n + 1), PROT_READ | PROT_WRITE, MAP_SHARED
| MAP_ANONYMOUS, -1, 0);
    pid = fork();
    if (pid == 0)
```

```c
    {
        // Tiến trình con
        int i = 0;
        while (n != 1)
        {
            buffer[i++] = n;
            if (n % 2 == 0)
                n /= 2;
            else
                n = 3 * n + 1;
        }
        buffer[i] = n;       // Thêm số cuối cùng (1) vào chuỗi
        buffer[i + 1] = -1; // Đánh dấu kết thúc chuỗi
    }
    else
    {
        // Tiến trình cha
        wait(NULL); // Đợi tiến trình con kết thúc

        // In chuỗi
        for (int i = 0; buffer[i] != -1; i++)
            printf("%d ", buffer[i]);
        printf("\n");

        // Dọn dẹp bộ nhớ chia sẻ
        munmap(buffer, sizeof(int) * (n + 1));
    }
    return 0;
}
```

```
ngocanh-22520077@TNANH-D53R8MB9:~$ gcc lab3_ontap.c -o collatz
ngocanh-22520077@TNANH-D53R8MB9:~$ ./collatz 8
8 4 2 1
ngocanh-22520077@TNANH-D53R8MB9:~$
```