

Họ và tên: Đỗ Phương Duy
Mã số sinh viên: 23520362
Lớp: KHTN2023

HỆ ĐIỀU HÀNH BÁO CÁO LAB 3

CHECKLIST (Đánh dấu x khi hoàn thành)

Lưu ý mỗi câu phải làm đủ 3 yêu cầu

I. Bài tập thực hành

	BT 1	BT 2	BT 3	BT 4
Trình bày cách làm	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Chụp hình minh chứng	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Giải thích kết quả	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

II. Bài tập ôn tập

	a
Trình bày cách làm	<input checked="" type="checkbox"/>
Chụp hình minh chứng	<input checked="" type="checkbox"/>
Giải thích kết quả	<input checked="" type="checkbox"/>

Tư chấm điểm: 10

*Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:
<MSV>_LABx.pdf

I. Bài tập thực hành:

1.

A. Ví dụ 3-1:

Code:

```
test_fork.c — dpduy123 [SSH: 192.168.31.47]

C hello.c 2 C test_fork.c 6 ×

IT007 > LAB3 > C test_fork.c > main(int, char * [])

1  /*#####
2  # University of Information Technology
3  # IT007 Operating System
4  #
5  # Do Phuong Duy, 23520362
6  # File: test_fork.c
7  #
8 #####
9 #include <stdio.h>
10 #include <stdlib.h>
11 #include <unistd.h>
12 #include <sys/wait.h>
13 #include <sys/types.h>
14 int main(int argc, char *argv[])
15 {
16     __pid_t pid;
17     pid = fork();
18     if (pid > 0)
19     {
20         printf("PARENTS | PID = %ld | PPID = %ld\n", (long)getpid(), (long)getppid());
21         if (argc > 2)
22             printf("PARENTS | There are %d arguments\n", argc - 1);
23         wait(NULL);
24     }
25     if (pid == 0)
26     {
27         printf("CHILDREN | PID = %ld | PPID = %ld\n", (long)getpid(), (long)getppid());
28         printf("CHILDREN | List of arguments: \n");
29         for (int i = 1; i < argc; i++)
30         {
31             printf("%s\n", argv[i]);
32         }
33     }
34     exit(0);
35 }
```

Giải thích code:

Hàm main nhận số lượng tham số argc và mảng các tham số (argv[])
__pid_t pid và pid = fork() là lệnh tạo tiến trình con.

Fork() tạo ra tiến trình con, gọi X là giá trị hàm này trả về. Nếu:

- X > 0 là tiến trình cha (PID của tiến trình con)
- X = 0 là tiến trình con
- X < 0 là đã có lỗi xảy ra

Nếu pid lớn hơn 0, nghĩa là đoạn mã này đang chạy trong tiến trình cha.

Nó in ra PID (Process ID) và PPID (Parent Process ID) của tiến trình cha.

Nếu có nhiều hơn 2 tham số (tính cả tên chương trình), nó sẽ in ra số lượng tham số.

Cuối cùng, tiến trình cha gọi wait(NULL) để chờ tiến trình con hoàn thành trước khi tiếp tục.

Nếu pid bằng 0, đoạn mã này đang chạy trong tiến trình con.

Chương trình in ra PID và PPID của tiến trình con.

Sau đó, in ra danh sách các tham số được truyền vào từ argv, bắt đầu từ argv[1] (bỏ qua tên chương trình ở argv[0]).

Biên dịch:

```
● dpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ gcc test_fork.c -o test_fork
● dpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ ./test_fork 128 64 32
PARENTS | PID = 6777 | PPID = 4163
PARENTS | There are 3 arguments
CHILDREN | PID = 6778 | PPID = 6777
CHILDREN | List of arguments:
128
64
32
```

Giải thích: biên dịch code trên bằng lệnh `./test fork 128 64 32` với 3 tham số là 128 64 32.

PID và PPID của tiến trình cha lần lượt là 6777 và 4163

PID và PPID của tiến trình con lần lượt là 6778 và 6777

B. Ví du 3-2:

`execl("./count.sh", "./count.sh", "10", NULL);` Thực thi tập lệnh shell count.sh với tham số "10". Nếu execl thành công, mã phía dưới sẽ không được thực thi.

Code:

The screenshot shows a macOS desktop environment with several open windows. In the foreground, a terminal window titled "test_exec1.c - dpduy123 [SSH: 192.168.31.47]" is displayed, showing the source code for a C program named "test_exec1.c". The code uses fork() and exec() to demonstrate process creation and execution. Below the terminal is a code editor window for the same file, "test_exec1.c", which also contains the same code. The code editor has tabs for "hello.c", "test_fork.c", "count.sh", and "test_exec1.c". The sidebar on the left shows a file tree with various files and folders, including "hello.c", "test_fork.c", "count.sh", and "test_exec1.c" under the "IT007/LAB3" folder. The status bar at the bottom indicates "SSH: 192.168.31.47".

```
Code File Edit Selection View Go Run Terminal Window Help
test_exec1.c - dpduy123 [SSH: 192.168.31.47]
EXPLORER OPEN EDITORS
hello.c test_fork.c count.sh test_exec1.c 6
IT007 > LAB3 > C test_exec1.c > main(int, char **)
1 // University of Information Technology
2 # IT007 Operating System
3 #
4 #
5 # <Your name>, <your Student ID>
6 # File: test_exec1.c
7 #
8 #####
9 #include <stdio.h>
10 #include <stdlib.h>
11 #include <unistd.h>
12 #include <sys/wait.h>
13 #include <sys/types.h>
14 int main(int argc, char* argv[])
15 {
16     _pid_t pid;
17     pid = fork();
18     if (pid > 0)
19     {
20         printf("PARENTS | PID = %ld | PPID = %ld\n", (long)getpid(), (long)getppid());
21         if (argc > 2)
22             printf("PARENTS | There are %d arguments\n", argc - 1);
23         wait(NULL);
24     }
25     if (pid == 0)
26     {
27         exec("./count.sh", "./count.sh", "10", NULL);
28         printf("CHILDREN | PID = %ld | PPID = %ld\n", (long)getpid(), (long)getppid());
29         printf("CHILDREN | List of arguments: \n");
30         for (int i = 1; i < argc; i++)
31         {
32             printf("%s\n", argv[i]);
33         }
34     }
35     exit(0);
36 }
```

Biên dịch:

```
● d duy123@dp duy123-VirtualBox:~/IT007/LAB3$ gcc test_execl.c -o test_execl
● d duy123@dp duy123-VirtualBox:~/IT007/LAB3$ ./test_execl 3 6 2
PARENTS | PID = 8693 | PPID = 7348
PARENTS | There are 3 arguments
Implementing: ./count.sh
PPID of count.sh:
dp duy123    8694    8693  0 00:05 pts/5    00:00:00 /bin/bash ./count.sh 10
dp duy123    8696    8694  0 00:05 pts/5    00:00:00 grep count.sh
```

Chương trình này minh họa việc tạo và xử lý tiến trình con trong C. Tiến trình cha in ra thông tin về nó và chờ tiến trình con. Tiến trình con thực thi một tập lệnh shell với tham số, và nếu việc thực thi thành công, nó sẽ không thực hiện các lệnh phía sau `exec()`. Chương trình sử dụng `fork()` để phân chia tiến trình và `exec()` để thay thế tiến trình con bằng một chương trình khác.

C. Ví dụ 3-3:

Code:

```
test_system.c - dp duy123 [SSH: 192.168.31.47]
1 ##### 
2 # University of Information Technology
3 # IT007 Operating System
4 #
5 # Do Phuong Duy, 23520362
6 # File: test_system.c
7 #
8 #####
9 #include <stdio.h>
10 #include <stdlib.h>
11 #include <unistd.h>
12 #include <sys/wait.h>
13 #include <sys/types.h>
14 int main(int argc, char* argv[])
15 {
16     printf("PARENTS | PID = %ld | PPID = %ld\n", (long)getpid(), (long)getppid());
17     if (argc > 2)
18         printf("PARENTS | There are %d arguments\n", argc - 1);
19     system("./count.sh 10");
20     printf("PARENTS | List of arguments: \n");
21     for (int i = 1; i < argc; i++)
22     {
23         printf("%s\n", argv[i]);
24     }
25     exit(0);
26 }
```

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Thân Thế Tùng.

Biên dịch:

```
● dpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ gcc test_system.c -o test_system
● dpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ ./test_system
PARENTS | PID = 3829 | PPID = 3650
Implementing: ./count.sh
PPID of count.sh:
dpduy123    3830    3829  0 21:23 pts/6    00:00:00 sh -c -- ./count.sh 10
dpduy123    3831    3830  0 21:23 pts/6    00:00:00 /bin/bash ./count.sh 10
dpduy123    3833    3831  0 21:23 pts/6    00:00:00 grep count.sh
PARENTS | List of arguments:
```

Giải thích:

Chương trình này minh họa cách sử dụng hàm `system()` để thực thi một lệnh shell từ trong một chương trình C. Nó in ra thông tin về PID và PPID, kiểm tra số lượng tham số, và sau đó thực thi tập lệnh `count.sh` với tham số "10". Cuối cùng, nó in ra danh sách các tham số đã truyền vào.

D. Ví dụ 3-4:

Process A khởi tạo bộ nhớ chia sẻ, ghi vào bộ nhớ "Hello Process B", sau đó chờ bộ nhớ được cập nhật bởi Process B.

Process B truy cập bộ nhớ chia sẻ, đọc dữ liệu do Process A ghi vào, sau đó cập nhật bộ nhớ với chuỗi "Hello Process A"

Code tiến trình A:

Code tiến trình B:

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Thân Thế Tùng.

The screenshot shows a macOS desktop environment. In the foreground, a terminal window is open with the command `./test_shm_a` running. The output shows three instances of the message "Waiting Process B update shared memory". In the background, a code editor (Xcode) is open, displaying the file `test_shm_b.c`. The code implements a shared memory application using the `shm_open`, `mmap`, and `strcpy` functions. The Xcode interface includes an Explorer sidebar with various source files and a timeline view.

Chạy process A:

```
○ dpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ ./test_shm_a
Waiting Process B update shared memory
Waiting Process B update shared memory
Waiting Process B update shared memory
```

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 muôn đo thời gian thực thi.

Code:

```
IT007 > LAB3 > C time.c > main(int, char * [])
● 1 ~ #include <stdio.h>
  2 ~ #include <stdlib.h>
  3 ~ #include <unistd.h>
  4 ~ #include <sys/wait.h>
  5 ~ #include <sys/types.h>
  6 ~ #include <sys/time.h>
  7
  8 ~ int main(int argc, char *argv[])
  9 ~ {
10 ~     // Kiểm tra số lượng tham số
11 ~     if (argc < 2) {
12 ~         fprintf(stderr, "Usage: %s <command>\n", argv[0]);
13 ~         return 1;
14 ~     }
15 ~     __pid_t pid;
16 ~     pid = fork();
17 ~     if (pid < 0) {
18 ~         // Nếu fork thất bại
19 ~         perror("Fork Failed");
20 ~         return 1;
21 ~     }
22 ~     if (pid > 0) {
23 ~         // Tiến Trình Cha
24 ~         int status;
25 ~         wait(&status);
26 ~         struct timeval start, end;
27 ~         gettimeofday(&end, NULL); // Lấy thời gian sau khi thực thi lệnh
28
29 ~         // Tính toán thời gian thực thi
30 ~         double elapsedTime = (end.tv_sec - start.tv_sec) +
31 ~                         (end.tv_usec - start.tv_usec) / 1000000.0;
32
33 ~         // In ra thời gian thực thi
34 ~         printf("Thời gian thực thi: %.5f giây\n", elapsedTime);
35 ~ }
```

```
36     else {
37         // Tiến trình con
38         struct timeval start, end;
39
40         // Lấy thời gian trước khi thực thi lệnh
41         gettimeofday(&start, NULL);
42
43         // Thực thi lệnh
44         execvp(argv[1], &argv[1]);
45
46         // Nếu execvp thất bại
47         perror("Exec failed");
48         exit(1);
49     }
50
51 }
```

Giải thích code:

1. Thư viện bao gồm:

- o stdio.h: Để sử dụng các hàm nhập/xuất.
- o stdlib.h: Để sử dụng các hàm chung như exit().
- o unistd.h: Để sử dụng fork() và exec().
- o sys/types.h và sys/wait.h: Để làm việc với tiến trình.
- o sys/time.h: Để sử dụng gettimeofday().

2. Kiểm tra tham số đầu vào:

- o Nếu số lượng tham số nhỏ hơn 2, in ra thông báo hướng dẫn và kết thúc chương trình.

3. Tạo tiến trình con:

- o Sử dụng fork() để tạo một tiến trình con.

4. Trong tiến trình con:

- o Sử dụng gettimeofday() để lấy thời gian bắt đầu.
- o Gọi execvp() để thực thi lệnh shell. execvp() cho phép bạn truyền danh sách các tham số.
- o Nếu execvp() thất bại, in ra thông báo lỗi và kết thúc tiến trình con.

5. Trong tiến trình cha:

- o Sử dụng wait() để chờ tiến trình con hoàn thành.
- o Sau khi tiến trình con kết thúc, lấy thời gian kết thúc bằng gettimeofday().
- o Tính toán thời gian thực thi bằng cách lấy hiệu giữa thời gian kết thúc và thời gian bắt đầu.
- o In ra thời gian thực thi.

Biên dịch và thực thi:

```
d duy123@dpduy123-VirtualBox:~/IT007/LAB3$ gcc time.c -o time
d duy123@dpduy123-VirtualBox:~/IT007/LAB3$ ./time ls
31      count.sh  hello    test.cpp   test_execl  test_fork   test_shm_a   test_shm_b   test_system   time
collatz.c  count.txt  hello.c  test-execl  test_excl.c  test_fork.c  test_shm_a.c  test_shm_b.c  test_system.c  time.c
Thời gian thực thi: 1730299252.31717 giây
```

3.

Code:

```
IT007 > LAB3 > C btth3.c > main(int, char * [])
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5 #include <sys/types.h>
6
7 void handle_sigint() {
8     printf("count.sh has stopped");
9 }
10 int main(int argc, char* argv[])
11 {
12     printf("Welcome to IT007, I am 23520362\n");
13     signal(SIGINT, handle_sigint);
14
15     __pid_t pid;
16     pid = fork();
17     if (pid > 0)
18     {
19         wait(NULL);
20     }
21     if (pid == 0)
22     {
23         execl("./count.sh", "./count.sh", "120", NULL);
24         //printf("CHILDREN | PID = %ld | PPID = %ld\n", (long)getpid(), (long)getppid());
25         //printf("CHILDREN | List of arguments: \n");
26         for (int i = 1; i < argc; i++)
27         {
28             printf("%s\n", argv[i]);
29         }
30     }
31     return 0;
32 }
```

Biên dịch:

```
● dpduy123@dpduy123-VirtualBox:~$ cd IT007/LAB3/
● dpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ ./btth3
Welcome to IT007, I am 23520362
Implementing: ./count.sh
PPID of count.sh:
dpduy123      5323      5322  0 21:56 pts/8    00:00:00 /bin/bash ./count.sh 120
dpduy123      5325      5323  0 21:56 pts/8    00:00:00 grep count.sh
1
2
3
4
5
○ ^Ccount.sh has stoppeddpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ █
```

Ở đây vì thời gian có hạn nên em Ctrl C khi count đếm đến 5

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

Code:

```
IT007 > LAB3 > C btth4.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/ipc.h>
6 #include <sys/shm.h>
7 #include <sys/wait.h>
8 #include <string.h>
9 #include <time.h>
10
11 #define BUFFER_SIZE 10 // Kích thước buffer
12 #define MAX_RANDOM 20
13 #define MIN_RANDOM 10
14
15 // Cấu trúc cho buffer
16 typedef struct {
17     int buffer[BUFFER_SIZE];
18     int count; // Số lượng phần tử hiện có trong buffer
19 } SharedBuffer;
20
21 int main() {
22     int shmid;
23     SharedBuffer *shared_buffer;
24
25     // Tạo bộ nhớ chia sẻ
26     shmid = shmget(IPC_PRIVATE, sizeof(SharedBuffer), IPC_CREAT | 0666);
27     if (shmid < 0) {
28         perror("shmget");
29         exit(1);
30     }
31
32     shared_buffer = (SharedBuffer *)shmat(shmid, NULL, 0);
33     if (shared_buffer == (SharedBuffer *) -1) {
34         perror("shmat");
35         exit(1);
36     }
37
38     shared_buffer->count = 0; // Khởi tạo số lượng phần tử
```

```
39     pid_t pid = fork(); // Tạo tiến trình con
40
41     if (pid < 0) {
42         perror("Fork failed");
43         exit(1);
44     } else if (pid == 0) {
45         // Tiến trình con - Consumer
46         int total = 0;
47
48         while (1) {
49             // Chờ cho có phần tử trong buffer
50             while (shared_buffer->count == 0) {
51                 usleep(100); // Ngủ 100 micro giây
52             }
53
54             // Đọc dữ liệu từ buffer
55             int value = shared_buffer->buffer[--shared_buffer->count];
56             printf("Consumer: Read value %d\n", value);
57             total += value;
58
59             // Kiểm tra tổng
60             if (total > 100) {
61                 printf("Consumer: Total is %d, stopping...\n", total);
62                 break;
63             }
64         }
65     }
66
67     // Giải phóng bộ nhớ chia sẻ và kết thúc tiến trình con
68     shmdt(shared_buffer);
69     exit(0);
70 } else {
71     // Tiến trình cha - Producer
72     srand(time(NULL));
73
74     while (1) {
75         // Tạo số ngẫu nhiên trong khoảng [10, 20]
76         int value = (rand() % (MAX_RANDOM - MIN_RANDOM + 1)) + MIN_RANDOM;
```

```
77     // Chờ cho có chỗ trong buffer
78     while (shared_buffer->count == BUFFER_SIZE) {
79         usleep(100); // Ngủ 100 micro giây
80     }
81
82     // Ghi dữ liệu vào buffer
83     shared_buffer->buffer[shared_buffer->count++] = value;
84     printf("Producer: Produced value %d\n", value);
85
86     // Kiểm tra nếu Consumer đã dừng
87     if (shared_buffer->count == BUFFER_SIZE && total > 100) {
88         break;
89     }
90 }
91
92     // Chờ tiến trình con kết thúc
93     wait(NULL);
94
95     // Giải phóng bộ nhớ chia sẻ
96     shmdt(shared_buffer);
97     shmctl(shmid, IPC_RMID, NULL);
98 }
99
100
101     return 0;
102 }
103 }
```

II. Bài tập ôn tập:

```
IT007 > LAB3 > C collatz.c > ⌂ is_positive_integer(const char *)
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <unistd.h>
4   #include <sys/types.h>
5   #include <sys/ipc.h>
6   #include <sys/shm.h>
7   #include <string.h>
8   #include <ctype.h>
9
10  #define BUFFER_SIZE 1024
11
12  // Hàm kiểm tra xem đầu vào có phải là số nguyên dương hay không
13  int is_positive_integer(const char *str) {
14      if (*str == '\0') return 0; // Rỗng
15      while (*str) {
16          if (!isdigit(*str)) return 0; // Không phải số
17          str++;
18      }
19      return 1; // Là số nguyên dương
20  }
21
22  int main(int argc, char *argv[]) {
23      if (argc != 2 || !is_positive_integer(argv[1])) {
24          fprintf(stderr, "Usage: %s <positive_integer>\n", argv[0]);
25          return 1;
26      }
27
28      int n = atoi(argv[1]); // Chuyển đổi chuỗi sang số nguyên
29      int shmid;
```

```
30     char *buffer;
31
32     // Tạo bộ nhớ chia sẻ
33     shmid = shmget(IPC_PRIVATE, BUFFER_SIZE, IPC_CREAT | 0666);
34     if (shmid < 0) {
35         perror("shmget");
36         return 1;
37     }
38
39     buffer = shmat(shmid, NULL, 0); // Gán bộ nhớ chia sẻ
40     if (buffer == (char *) -1) {
41         perror("shmat");
42         return 1;
43     }
44
45     pid_t pid = fork(); // Tạo tiến trình con
46
47     if (pid < 0) {
48         perror("Fork failed");
49         return 1;
50     } else if (pid == 0) {
51         // Tiến trình con
52         char temp[32];
53         int index = 0;
54
55         while (n != 1) {
56             index += sprintf(&buffer[index], "%d, ", n); // Ghi số vào buffer
57             if (n % 2 == 0) {
58                 n /= 2;
```

```
59     } else {
60         n = 3 * n + 1;
61     }
62 }
63 sprintf(&buffer[index], "1"); // Thêm 1 vào buffer
64
65 // Kết thúc tiến trình con
66 shmdt(buffer); // Ngắt kết nối bộ nhớ chia sẻ
67 exit(0);
68 } else {
69     // Tiến trình cha
70     wait(NULL); // Chờ tiến trình con hoàn thành
71
72     // In kết quả ra màn hình
73     printf("Collatz sequence: %s\n", buffer);
74
75     // Giải phóng bộ nhớ chia sẻ
76     shmdt(buffer);
77     shmctl(shmid, IPC_RMID, NULL);
78 }
79
80 return 0;
81 }
82 }
```

Giải thích code:

- **Hàm kiểm tra số nguyên dương:**
 - Hàm `is_positive_integer()` kiểm tra xem chuỗi đầu vào có phải là số nguyên dương không. Nếu không, chương trình sẽ in thông báo lỗi và thoát.
- **Bộ nhớ chia sẻ:**
 - Tạo bộ nhớ chia sẻ bằng `shmget()` và gán nó vào buffer sử dụng `shmat()`.
- **Tiến trình con:**
 - Tiến trình con tính toán chuỗi Collatz từ số `n`. Mỗi giá trị được ghi vào buffer sử dụng `sprintf()`.
 - Khi `n` trở về 1, tiến trình con kết thúc và ngắt kết nối bộ nhớ chia sẻ.
- **Tiến trình cha:**
 - Tiến trình cha chờ cho tiến trình con hoàn thành bằng `wait()`, sau đó in ra chuỗi Collatz từ buffer.
 - Giải phóng bộ nhớ chia sẻ bằng `shmdt()` và `shmctl()`.

Biên dịch và kiểm thử các testcase:

```
● dpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ ./collatz 8
Collatz sequence: 8, 4, 2, 1
● dpduy123@dpduy123-VirtualBox:~/IT007/LAB3$ ./collatz 35
Collatz sequence: 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1
```