Bài thực hành số 4

Lập trình shared memory

Ghi chú:

- Sinh viên nộp bài tại trang web: `www.cse.hcmut.edu.vn/portal`
- File nộp bài đặt tên là: *`lab4.tar.bz2`* (sử dụng lệnh *make pack*)
- Hạn chót nộp bài: *`11:59PM 12/08/2010`*
- SV có thể chỉnh sửa source code mẫu nếu thấy cần thiết
- Mọi gian lận sẽ nhận điểm KHÔNG nếu bị phát hiện

1 Giới thiệu

1.1 Mục tiêu

Viết một chương trình C trên Linux để tính tổng các thừa số nguyên tố của các số nguyên sử dụng kỹ thuật lập trình shared memory và semaphore.

1.2 Kiến thức cần biết

- Lập trình C trên Linux
- Lập trình multiprocess
- Lập trình semaphore
- Lập trình shared memory
- Makefile

1.3 Mô tả chương trình

Đầu vào (Input) của chương trình:

- Các số nguyên

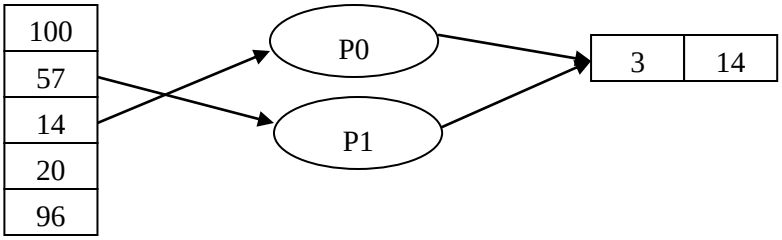
Đầu ra (Output) của chương trình:

- Tổng tất cả các thừa số nguyên tố của tất cả các số nguyên đó.

Hiện thực chương trình trên bằng cách sử dụng vùng nhớ chia sẻ (shared memory) để chia sẻ thông tin giữa các quá trình và sử dụng semaphore để giải quyết các vấn đề đụng độ trên vùng nhớ chia sẻ đó (2 process không thể đồng thời ghi dữ liệu lên vùng nhớ chia sẻ).

1.4 Ý tưởng hiện thực chương trình

- Khi chạy, chương trình sử dụng hàm fork để tạo 2 process mới: process cha và process con.
- Hai process này hoạt động theo mô hình work pool (minh họa ở hình vẽ bên dưới).



- Các thông số dòng lệnh được lưu vào một mảng (2 process cha và con đều có thể truy xuất được mảng này). Sau đó process cha và con sử dụng vùng nhớ chia sẻ để lưu thông tin chia sẻ.
- Vùng nhớ chia sẻ gồm có 2 số nguyên, số đầu tiên mô tả chỉ số tiếp theo của mảng sẽ được xử lý, số nguyên thứ hai mô tả kết quả tính toán đến thời điểm hiện tại.
- Khi bắt đầu xử lý một phần tử, process đọc số nguyên đầu tiên trong vùng nhớ chia sẻ để lấy chỉ số của phần tử cần xử lý, sau đó process này tăng chỉ số này lên và lưu giá trị chỉ số mới vào vùng nhớ chia sẻ.
- Tiếp theo process đó lấy được phần tử cần xử lý từ mảng, xử lý phần tử đó và tính được tổng của các thừa số nguyên tố của số nguyên đó.
- Sau đó, process cộng dồn tổng này vào tổng toàn phần (giá trị của số nguyên thứ 2 trong vùng nhớ chia sẻ).
- Process nào xử lý sau cùng sẽ in giá trị tổng toàn phần ra màn hình.
- Để giải quyết tranh chấp trên vùng nhớ chia sẻ, ta sử dụng một biến semaphore để đồng bộ quá trình đọc và ghi vào vùng nhớ này.

1.5 Một số kỹ thuật lập trình

1.5.1 Xử lý tham số dòng lệnh

Khi chạy chương trình người dùng có thể nhập vào nhiều tham số, để lấy được chính xác các tham số này, ta có thể xử lý trên hai tham số *argc* & *argv* của hàm *main*:

```
int main(int argc, char* argv[])
```

Với cách khai báo như trên, agrc chứa số tham số mà người dùng nhập trên dòng lệnh (kể cả tên chương trình chạy) và tham số argv chứa các chuỗi tương ứng với các tham số đó.

Ví dụ khi ta chạy lệnh:

```
sum 2 5 7 9
```

để tính tổng các số ta truyền vào từ dòng lệnh, thì tham số `argc=5` và `argv={"sum","2","5","7","9"}`.

1.5.2 Kỹ thuật lập trình shared memory

Tham khảo slide trên lớp

1.5.3 Kỹ thuật lập trình semaphore

Tham khảo slide trên lớp

2 Yêu cầu

Cú pháp chương trình:

```
sumprime [số nguyên dương]+
```

Tham số của chương trình là các số nguyên dương, và chương trình phải có ít nhất một tham số để tính toán. Kết quả tính toán được in ra màn hình.

3 Ví dụ

```
$ ./sumprime 3
3
$ ./sumprime 6
5
$ ./sumprime 8
6
$ ./sumprime 3 6 8
14
```