Bài tập lớn số 1

Giả lập bộ định thời

Ghi chú:

- Sinh viên nộp bài tại trang web: `www.cse.hcmut.edu.vn/portal`
- File nộp bài đặt tên là: *`ass1.tar.bz2`*
- Hạn chót nộp bài: *`08:00am 10/08/2010`*
- SV có thể chỉnh sửa source code mẫu nếu thấy cần thiết
- Mọi gian lận sẽ nhận điểm 0 (KHÔNG) nếu bị phát hiện
- Mỗi nhóm 3 sinh viên (SV), mã nguồn cả 'project' chỉ là một và phải biên dịch được.

1 Giới thiệu

1.1 Mục tiêu

Hãy viết một chương trình C trên Linux để **giả lập các giải thuật định thời** đã được học như: Shortest Time First (SJF), Round-Robin. Giả thuyết:

- Giả sử hệ thống có p (p >= 1) bộ xử lý.
- Nếu có *N* process trong hàng đợi ready, và quantum time là *q*

1.2 Kiến thức cần biết

- Các giải thuật định thời
- Lập trình C trên Linux
- Lập trình Shell
- Makefile

1.3 Mô tả chương trình

Đầu vào (Input) của chương trình:

- Bộ định thời: FCFS, SRTF, SJF, RR, ...
- Đặc tả của các quá trình trong hệ thống, chẳng hạn: thời gian vào hệ thống, thời gian thực thi của quá trình. Ngoài ra, tuỳ vào từng bộ định thời mà ta xây dựng thêm các đặc tả khác như: mức độ ưu tiên, quantum time, ...

Đầu ra (Output) của chương trình:

- Ghi lại thông tin định thời các quá trình trong hệ thống tại từng thời điểm, thời gian còn thực thi còn lại của quá trình đó.

1.4 Chương trình sườn

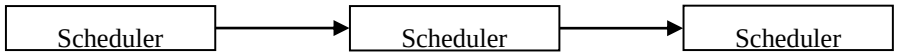
1.4.1 Cấu trúc chương trình sườn

Gồm có 3 phần: `core, scheds, tasks`. Mỗi phần tương ứng với một thư mục:

- Phần core**:** Chứa thông tin đặc tả những cấu trúc được dùng chung trong việc hiện thực các bộ định thời. Phần core chứa những đoạn code hầu như không bị thay đổi khi thêm hay bớt một bộ định thời.
- Phần scheds**:** Phần hiện thực những chức năng chính của bộ định thời. Trong thư mục *scheds*, phần hiện thực những bộ định thời khác nhau được đặt trong những thư mục khác nhau.
- Phần tasks: Phần hiện thực chức năng chuyển thông tin từ file input thành thông tin của các task. Vì các bộ định thời khác nhau có cấu trúc file input và thông tin về task cũng khác nhau, do đó các bộ định thời khác nhau nên được hiện thực trong những thư mục khác nhau.

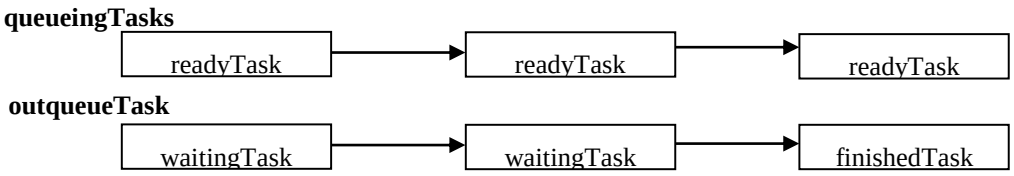
1.4.2 Ý tưởng hiện thực chương trình

Tại một thời điểm, chương trình cần phải nạp vào tất cả các bộ định thời để người dùng sử dụng. Để quản lý các bộ định thời này, ta sử dụng danh sách các bộ định thời. Trong đó chỉ có duy nhất 1 bộ định thời ở trạng thái *active*, các bộ định thời còn lại ở trạng thái *inactive*. Khi chạy chương trình, người dùng chỉ có thể tương tác trên bộ định thời *active* mà thôi.



Bộ định thời chứa danh sách tất cả các task trong hệ thống. Các task này được chia ra làm hai danh sách:

- `queueingTasks`: chứa danh sách các task đã sẵn sàng để được định thời. Đây là những task thực sự đi vào hệ thống.
- `outqueueTasks`: chứa danh sách các tasks chưa đi vào hệ thống hoặc các task đã thực thi xong.



Đầu tiên, scheduler đọc file input để lấy thông tin tất cả các task và đặt nó vào *outqueueTasks*. Mỗi task trong *outqueueTasks* chứa thông tin về task đó, bao gồm *thời gian sẽ đi vào hệ thống* (khi đến thời gian này thì task này được chuyển sang danh sách *queueingTasks*), *thời gian cần thiết để task này thực thi xong*, *trạng thái hiện tại của task này*, ... Khi vừa được tạo thì các task có trạng thái là *chưa bắt đầu* và nằm trong *outqueueTasks*.

Mỗi lần scheduler định thời các task, biến timer tăng lên 1, sau đó nó kiểm tra trong *outqueueTasks* xem có task nào đã sẵn sàng chưa (kiểm tra thời gian đi vào hệ thống của task đó), nếu có task sẵn sàng rồi thì chuyển task đó vào cuối danh sách *queueingTasks* và chuyển trạng thái của nó sang *đang ngủ*.

Lúc này *queueingTasks* chứa danh sách của các task sẵn sàng để thực thi. Tùy vào bộ định thời là gì mà scheduler chọn ra task nào trong *queueingTasks* để chạy. Với các bộ định thời khác nhau, ta sẽ có các giải thuật khác nhau để chọn task chạy tiếp theo.

Khi chọn được task để chạy rồi, chuyển task này sang trạng thái *đang chạy,* sau đó chạy task này. Sau đó kiểm tra task này chạy xong chưa, nếu đã chạy xong rồi thì chuyển task này vào cuối danh sách của *outqueueTasks*. Sau đó cập nhật lại trạng thái cho task này là *đã hoàn thành*.

Tại một thời điểm, có nhiều nhất là 1 task có trạng thái *đang chạy* trong *queueingTasks*.

Khi *queueingTasks* đã hết task để chạy và *outqueueTasks* đã hết task để định thời thì dừng công việc định thời. Kết thúc quá trình thực thi.

1.4.3 Sử dụng chương trình sườn

Để có thể hiện thực một bộ định thời khác (vẫn dùng chung phần tasks), ta chỉ cần thêm, viết lại code vào phần scheds, và có thể không cần thay đổi Makefile. Khi thêm mới một bộ định thời, ta cần chỉnh sửa một số phần sau đây:

- Hàm `collectSupportedSchedulers` trong file `schedsim.c` và thêm các định nghĩa cho bộ định thời mới trong `schedsim.h` và `sched_man.h`
- Sửa lại các file `task_man.h` và `task_man.c` trong `./tasks/new_sched/` cho phù hợp với bộ định thời mới (giả sử phần hiện thực bộ định thời mới được đặt trong thư mục `./scheds/new_sched/`)
- Sửa lại các file `sched.h` và `sched.c` trong `./scheds/new_sched/` (giả sử phần hiện thực bộ định thời mới được đặt trong thư mục `./scheds/new_sched/`)
- Chỉnh lại `Makefile` trong `./scheds/new_sched/` để xác định bộ xử lý quá trình nào được dùng trong phần `tasks`. Trong file Makefile này, cũng chỉ cần sửa lại thông tin biến `TASK_MAN_PATH`

1.5 Một số kỹ thuật lập trình

1.5.1 Xử lý thông số nhập vào từ chương trình

Một chương trình tốt thường cho phép người dùng thiết lập một vài thông số khi chạy chương trình, chẳng hạn khi thực hiện lệnh:

```
$ ls -R
```

người dùng đã truyền vào thông số `-R` để liệt kê các file và thư mục không chỉ trong thư mục hiện hành mà còn cả những thư mục con của thư mục hiện hành nếu có.

Sau đây là đoạn chương trình mẫu, sử dụng hàm `getopt()` để xử lý thông số -R ở trên:

```
int opt;
extern char *optarg;

while ((opt = getopt(argc, argv, "R")) != EOF) {
    switch (opt) {
        case 'R':
            // Option -R occurs
            // Proccess that option here
            break;
        default:
            // Other options
            break;
    }
}
```

1.5.2 Xử lý file cấu hình

File cấu hình thường ở dạng text và có cấu trúc (đơn giản). Ta thường sử dụng các hàm sau để xử lý những dạng file này:

- fopen: mở một file
- fclose: đóng file đã mở

- fscanf: đọc/lọc thông tin một dòng trong file
- fprintf: ghi thông tin lên file

1.5.3 Một số lệnh hữu ích trong shell

Shell là một trong những công cụ hỗ trợ hữu ích nhất trong Linux. Ngoài một số lệnh thông dụng như: `cd`, `mkdir`, `ls`, `top`, `...`, còn rất nhiều lệnh khác với đa dạng chức năng. Tuy nhiên, đối với chương trình giả lập định thời này, ta có thể tham khảo cách sử dụng của các lệnh sau:

- `sed`: xử lý chuỗi
- `awk/gawk`: ngôn ngữ lập trình AWK, có thể dùng được trong shell
- `for ... in ... ; do ... ; done`: Cấu trúc lặp

2 Yêu cầu

Hiện thực 2 giải thuật định thời là ***định thời xoay vòng*** **(round robin),** và ***định thời Shortest Job First (SJF)***.

Chương trình sau khi biên dịch có tên là `schedsim`**,** hỗ trợ các thông số sau:

- `-h`: Hiển thị thông tin hướng dẫn sử dụng chương trình
- `-s` *`id`*: Chọn lựa bộ định thời để giả lập. Mặc định chọn bộ định thời *round robin*
- `-i` *`filename`*: Chọn file input. Đây là file chứa thông tin các task sẽ được định thời. Tên file input mặc định là *`input.txt`*
- `-o` *`filename`*: Chọn file output. Đây là file chứa kết quả định thời của scheduler. Tên file output mặc định là *`output.txt`*
- `-l` : Liệt kê tất cả những bộ định thời được hỗ trợ bởi chương trình. Thông tin liệt kê gồm *số thứ tự*, *id bộ định thời*, *tên bộ định thời* và *phần mô tả*. Các thông tin này cách nhau bởi dấu TAB. Tham khảo thêm trong source code gửi kèm.

Cú pháp chạy chương trình `schedsim`:

```
schedsim [-h] [-s id] [-i filename] [-o filename] [-l]
```

Một số lưu ý khi xử lý option:

- Nếu dòng lệnh có option không nằm trong những option ở trên thì báo lỗi, thoát chương trình ngay lập tức và không quan tâm đến các option khác.
- Độ ưu tiên của option -h là cao nhất, sau đó mới đến option -l, sau đó là các option để chạy chương trình. Điều này có nghĩa là nếu dòng lệnh có cả 2 option -h và -l thì ta chỉ xử lý option -h, không quan tâm đến option -l. Tương tự như vậy, nếu dòng lệnh có option -l và các option chạy chương trình thì ta chỉ quan tâm đến option -l, và xem như không có các option còn lại.

3 Định dạng file input, output

3.1 Bài toán định thời xoay vòng (round-robin scheduling)

Định thời xoay vòng theo cơ chế preemptive.

Quantum time của bộ định thời được định nghĩa trong file ./scheds/rr/sched.conf là một số nguyên dương lớn hơn 0. Quantum time được tính theo đơn vị thời gian của bộ định thời.

Thông tin bộ định thời

ID: `3`
Tên : `rr`
Mô tả: `Round Robin scheduler`

File input

Mỗi dòng mô tả một quá trình, bao gồm *thời điểm đi vào hệ thống* (`start time`) và *thời gian thực thi* (`duration`). Các thông tin này cách nhau bởi một khoảng trắng. Ví dụ:

```
0 3
1 2
```

Quá trình 0 đi vào hệ thống tại thời điểm 0, thực thi trong 3 đơn vị thời gian. Quá trình 1 đi vào hệ thống tại thời điểm 1, thực thi trong 2 đơn vị thời gian.

File output

Mỗi dòng cần xuất ra các thông tin tại mỗi thời điểm: *thời điểm hiện tại*, *ID của quá trình đang thực thi*, *duration còn lại của quá trình đó*. Ví dụ, với file input như ở trên, **quantum time bằng 2** và với giải thuật định thời `RR`, file output sẽ như sau

```
0 0 2
1 0 1
2 1 1
3 1 0
4 0 0
5
```
`// Luôn có dòng cuối cùng này. Không có khoảng trắng sau số 5`

Trong trường hợp hệ thống tạm thời không có quá trình nào đang thực thi, chỉ cần ghi vào file output thông tin thời điểm (bỏ trống 2 thông tin `pid` và `duration`). Khi đó thông số thời gian trong file output sẽ tăng dần 0, 1, 2, … chứ không nhảy cách quãng. Chẳng hạn với file input:

```
0 1

2 2
```

thì ta sẽ được file output như sau:

```
0 0 0
1
2 1 1
3 1 0
4
```
`// Không có khoảng trắng sau số 1`

3.2 Bài toán định thời SJF

Bộ định thời theo giải thuật SJF.

Thông tin bộ định thời

ID: `5`
Tên: `sjf`
Mô tả: `Shortest Job First scheduler`

File input

Tương tự như bộ đình thời xoay vòng, ta có file input

```
0  3
1  2
2  1
```

Với file input mẫu ở trên, chương trình sẽ hiểu như sau: có 3 quá trình sẽ được thực thi trong giai đoạn giả lập. Quá trình đầu tiên đi vào hệ thống tại thời điểm 0 và thực thi trong vòng 3 đơn vị thời gian. Quá trình thứ hai đi vào hệ thống tại thời điểm 1 và thời gian cần để thực thi là 2 đơn vị thời gian. Quá trình thứ ba đi vào hệ thống tại thời điểm 2 và thời gian cần để thực thi là 1 đơn vị thời gian.

File output

Tương tự định dạng output của giải thuật định thời xoay vòng, ta có file output:

```
0  0  2
1  0  1
2  0  0
3  2  0
4  1  1
5  1  0
6
```

Tài liệu tham khảo

[1] *GCC Manuals*, http://gcc.gnu.org/onlinedocs/gcc-4.2.3/gcc/

[2] *C programming tutorial*, http://www.cprogramming.com/tutorial.html

[3] *C programming tutorial*, http://www.iu.hio.no/~mark/CTutorial/CTutorial.html

[5] *Tutorial – Make file*, http://www.opussoftware.com/tutorial/TutMakefile.htm

[6] *GNU Make*, http://www.gnu.org/software/make/manual/make.html

[7] *Man page*

[8] A. Silberschatz, P. B. Galvin, G. Gagne, 2006, *Operating System Principles, 7th Edition*, John Wiley & Sons Inc.