Đại học Bách khoa TP.HCM
Khoa Khoa học & Kỹ thuật Máy tính

Khóa học: Hệ điều hành
Bài tập - Hệ điều hành đơn giản

Ngày 7 tháng 11 năm 2022

Mục tiêu: Mục tiêu của bài tập này là mô phỏng các thành phần chính trong một hệ điều hành đơn giản, ví dụ: bộ lập lịch, đồng bộ hóa, các hoạt động liên quan của bộ nhớ vật lý và bộ nhớ ảo.

Nội dung: Cụ thể, sinh viên sẽ thực hành với 3 module chính: lập lịch, đồng bộ hóa, cơ chế cấp phát bộ nhớ từ bộ nhớ ảo sang bộ nhớ vật lý.

- Người lập kế hoạch
- đồng bộ hóa
- hoạt động phân bổ mem từ ảo sang vật lý

Kết quả: Sau bài này HS hiểu được phần nào nguyên lý hoạt động của một HĐH đơn giản. Các em rút ra được vai trò, ý nghĩa của các module chính trong HĐH cũ ng như cách thức hoạt động của nó.

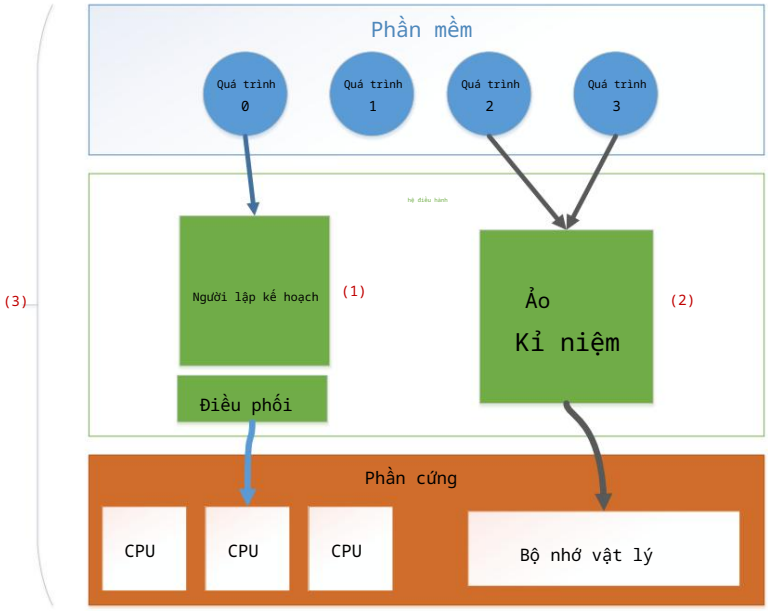
nội dung

1. Giới thiệu

1.1 Tổng quan

Bài tập mô phỏng một hệ điều hành đơn giản để giúp sinh viên hiểu các khái niệm cơ bản về lập lịch, đồng bộ hóa và quản lý bộ nhớ. Hình 1 cho thấy kiến trúc tổng thể của "hệ điều hành" mà chúng ta sắp triển khai. Nói chung, HĐH phải quản lý hai tài nguyên "ảo": (các) CPU và RAM bằng hai thành phần cốt lõi:

- Scheduler (and Dispatcher): xác định tiến trình được phép chạy trên CPU nào.
- Công cụ bộ nhớ ảo (VME): cô lập không gian bộ nhớ của từng tiến trình với nhau. RAM vật lý được chia sẻ bởi nhiều quy trình nhưng mỗi quy trình không biết sự tồn tại của quy trình khác. Điều này được thực hiện bằng cách cho phép mỗi quy trình có không gian bộ nhớ ảo riêng và công cụ bộ nhớ ảo sẽ ánh xạ và dịch địa chỉ ảo do quy trình cung cấp sang địa chỉ vật lý tương ứng.



Hình 1: Khung cảnh chung của các module chính trong nhiệm vụ này

Thông qua các mô-đun đó, Hệ điều hành cho phép nhiều quy trình do người dùng tạo chia sẻ và sử dụng tài nguyên máy tính "ảo". Do đó, trong nhiệm vụ này, chúng tôi tập trung vào việc triển khai bộ lập lịch/bộ điều phối và công cụ bộ nhớ ảo.

1.2 Mã nguồn

Sau khi tải mã nguồn của bài tập tại mục "Tài nguyên" trên nền tảng cổng thông tin và giải nén, bạn sẽ thấy mã nguồn được tổ chức như sau.

- Tệp tiêu đề
  - timer.h: Xác định bộ đếm thời gian cho toàn hệ thống.
  - cpu.h: Xác định các chức năng được sử dụng để triển khai CPU ảo

- queue.h: Các hàm được sử dụng để triển khai hàng đợi chứa PCB của các quy trình
- sched.h: Xác định các chức năng được sử dụng bởi bộ lập lịch
- mem.h: Các hàm được Virtual Memory Engine sử dụng.
- loader.h: Các chức năng được sử dụng bởi bộ nạp tải chương trình từ đĩa vào bộ nhớ
- common.h: Xác định các cấu trúc và chức năng được sử dụng ở mọi nơi trong HĐH.

• Tệp nguồn

- timer.c: Triển khai bộ đếm thời gian.
- cpu.c: Triển khai CPU ảo.
- queue.c: Thực hiện các hoạt động trên hàng đợi (mức độ ưu tiên).
- paging.c: Sử dụng để kiểm tra chức năng của Virtual Memory Engine.
- os.c: Toàn bộ HĐH bắt đầu chạy từ tệp này.
- loader.c: Triển khai bộ tải
- sched.c: Triển khai bộ lập lịch
- mem.c: Triển khai RAM và Công cụ bộ nhớ ảo

• Tạo tệp tin

• đầu vào Mẫu đầu vào được sử dụng để xác minh

• đầu ra Đầu ra mẫu của hệ điều hành.

1.3 Quy trình

Chúng tôi sẽ xây dựng một hệ điều hành đa nhiệm cho phép nhiều quy trình chạy đồng thời, do đó, đáng để dành một chút không gian để giải thích việc tổ chức các quy trình. Hệ điều hành quản lý các tiến trình thông qua PCB của chúng được mô tả như sau:

```
// Từ bao gồm/common.h struct pcb_t {

       uint32_t pid; ưu
       tiên uint32_t; uint32_t
       mã_seg_t * mã; đăng ký addr_t; uint32_t
       máy tính; cấu trúc seg_table_t *
       seg_table; uint32_t bp;



}
```

Ý nghĩa các trường trong cấu trúc:

• PID: PID của tiến trình

• priority: Mức độ ưu tiên của tiến trình, Bộ lập lịch sẽ cho phép các tiến trình có mức ưu tiên cao hơn chạy trước tiến trình có mức ưu tiên cao hơn ưu tiên thấp hơn.

• code: Đoạn văn bản của quy trình (Để đơn giản hóa việc mô phỏng, chúng tôi không đặt đoạn văn bản trong ĐẬP).

• res: Thanh ghi, mỗi tiến trình có thể sử dụng tối đa 10 thanh ghi được đánh số từ 0 đến 9.

- pc: Vị trí hiện tại của bộ đếm chương trình.
- seg table: Bảng trang dùng để dịch địa chỉ ảo sang địa chỉ vật lý.
- bp: Con trỏ ngắt, dùng để quản lý heap segment.

Tương tự như tiến trình thực, mỗi tiến trình trong mô phỏng này chỉ là một danh sách các lệnh được CPU thực hiện lần lượt từ đầu đến cuối (chúng ta không thực hiện các lệnh nhảy ở đây). Có năm hướng dẫn mà một quy trình có thể thực hiện:

- CALC: thực hiện một số tính toán bằng CPU. Hướng dẫn này không có đối số.
- ALLOC: Phân bổ một số đoạn byte trên bộ nhớ chính (RAM). Cú pháp của lệnh:

```
phân bổ [kích thước] [reg]
```

trong đó size là số byte mà tiến trình muốn cấp phát từ RAM và reg là số lượng thanh ghi sẽ lưu địa chỉ của byte đầu tiên của vùng bộ nhớ được cấp phát. Ví dụ: lệnh cấp phát 124 7 sẽ phân bổ 124 byte từ HĐH và địa chỉ của byte đầu tiên trong số 124 byte đó được lưu trữ tại thanh ghi số 7.

- MIỄN PHÍ Bộ nhớ được cấp phát miễn phí. Cú pháp:

```
miễn phí [reg]
```

trong đó reg là số thanh ghi chứa địa chỉ của byte đầu tiên của vùng bộ nhớ được giải phóng.

- READ Đọc một byte từ bộ nhớ. Cú pháp:

```
đọc [nguồn] [bù] [đích]
```

Lệnh đọc một byte bộ nhớ tại địa chỉ bằng với giá trị của thanh ghi nguồn + offset và lưu nó vào đích. Ví dụ, giả sử rằng giá trị của thanh ghi #1 là 0x123 thì lệnh đọc 1 20 2 sẽ đọc một byte bộ nhớ ở địa chỉ 0x123 + 14 (14 là 20 trong hệ thập lục phân) và lưu nó vào thanh ghi #2.

- WRITE Ghi một thanh ghi giá trị vào bộ nhớ. Cú pháp:

```
ghi [dữ liệu] [đích] [bù đắp]
```

Lệnh ghi dữ liệu vào địa chỉ bằng với giá trị của thanh ghi đích + offset. Ví dụ, giả sử rằng giá trị của thanh ghi #1 là 0x123 thì lệnh ghi 10 1 20 sẽ ghi 10 vào bộ nhớ ở địa chỉ 0x123 + 14 (14 là 20 trong hệ thập lục phân).

1.4 Làm thế nào để tạo một quy trình?

Nội dung của mỗi quá trình thực sự là một bản sao của một chương trình được lưu trữ trên đĩa. Vì vậy, để tạo ra một quy trình, trước tiên chúng ta phải tạo ra chương trình mô tả nội dung của nó. Một chương trình được xác định bởi một tệp duy nhất có định dạng sau:

```
[ưu tiên] [N = số lệnh]
hướng dẫn 0
hướng dẫn 1
...
hướng dẫn N-1
```

trong đó priority là mức độ ưu tiên mặc định của quy trình được tạo từ chương trình này. Mức độ ưu tiên càng cao thì cơ hội xử lý đó được CPU chọn từ hàng đợi càng cao (Xem phần 2.1 để biết thêm chi tiết). Hãy nhớ rằng giá trị mặc định này có thể được ghi đè bởi mức ưu tiên "trực tiếp" trong quá trình gọi thực thi quy trình.
Để giải quyết xung đột, khi nó có mức độ ưu tiên trong quá trình tải, nó sẽ ghi đè (hay còn gọi là được chọn) và thay thế mức độ ưu tiên mặc định trong tệp mô tả quy trình.
N là số lệnh và mỗi dòng trong số N dòng tiếp theo là các lệnh được trình bày theo định dạng đã đề cập trong phần trước. Bạn có thể mở các tệp trong thư mục input/proc để xem một số mẫu

các chương trình.

1.5 Cách chạy mô phỏng

Những gì chúng ta sẽ làm trong nhiệm vụ này là triển khai một hệ điều hành đơn giản và mô phỏng nó trên phần cứng ảo. Để bắt đầu quá trình mô phỏng, trước tiên chúng ta phải mô tả cấu hình của phần cứng và môi trường mà chúng ta sẽ mô phỏng. Điều này được thực hiện bằng cách tạo một tệp cấu hình ở định dạng sau

```
[lát thời gian] [N = Số lượng CPU] [M = Số lượng quy trình được chạy]
[thời gian 0] [đường dẫn 0] [mức độ ưu tiên
0] [thời gian 1] [đường dẫn 1] [mức độ ưu tiên 1]
...
[thời gian M-1] [đường dẫn M-1] [ưu tiên M-1]
```

trong đó lát cắt thời gian là khoảng thời gian mà một quy trình được phép chạy. N là số lượng CPU có sẵn và M là số lượng quy trình sẽ được chạy. Tham số ưu tiên cuối cùng là ưu tiên "trực tiếp" khi quy trình được gọi và điều này sẽ ghi đè lên ưu tiên mặc định trong tệp mô tả quy trình (tham khảo phần 1.4). Mỗi dòng trong M dòng tiếp theo là thời điểm bắt đầu một tiến trình và đường dẫn đến tệp chứa nội dung của chương trình sẽ được tải. Bạn có thể tìm thấy các tập tin cấu hình tại thư mục đầu vào.

Để bắt đầu mô phỏng, trước tiên bạn phải biên dịch mã nguồn bằng cách sử dụng lệnh Make all. Sau đó, chạy lệnh

```
./os [đường dẫn cấu hình]
```

trong đó đường dẫn cấu hình là đường dẫn để định cấu hình tệp cho môi trường mà bạn muốn chạy trên đó.

2 Thực hiện

2.1 Bộ lập lịch

Đầu tiên chúng tôi thực hiện lịch trình. Hình 2 cho thấy cách hệ điều hành lên lịch xử lý. Mặc dù HĐH được thiết kế để hoạt động trên nhiều bộ xử lý, nhưng trong phần bài tập này, chúng tôi giả sử hệ thống có nhiều bộ xử lý. Hệ điều hành sử dụng hệ thống hàng đợi sẵn sàng để xác định quy trình nào sẽ được thực thi khi CPU khả dụng. Thiết kế bộ lập lịch dựa trên thuật toán "hàng đợi phản hồi đa cấp" được sử dụng trong nhân Linux.

Theo hình 2, bộ lập lịch hoạt động như sau. Đối với mỗi chương trình mới, trình tải sẽ tạo một quy trình mới và gán một PCB mới cho nó. Sau đó, bộ nạp sẽ đọc và sao chép nội dung của chương trình vào đoạn văn bản của quy trình mới (được trỏ bởi con trỏ mã trong PCB của quy trình - mục 1.3). Cuối cùng, PCB của tiến trình được đẩy vào hàng đợi sẵn sàng và đợi CPU. CPU chạy các tiến trình theo kiểu quay vòng. Mỗi quá trình được phép chạy trong một khoảng thời gian nhất định. Sau đó, CPU buộc phải đưa tiến trình vào hàng đợi sẵn sàng ưu tiên thích hợp. Sau đó, CPU chọn một tiến trình khác từ hàng đợi sẵn sàng và tiếp tục chạy. Phần còn lại của luồng thuộc về mã quản lý CPU để thực thi nó trong hàng đợi sẵn sàng.

Trong hệ thống này, chúng tôi triển khai chính sách Hàng đợi đa cấp (MLQ). Hệ thống chứa các mức ưu tiên MAX PRIO. Mặc dù hệ thống thực, tức là nhân Linux, có thể nhóm các mức này thành tập hợp con, chúng tôi vẫn giữ thiết kế trong đó mỗi mức ưu tiên được giữ bởi một hàng đợi sẵn sàng để đơn giản. Chúng tôi đơn giản hóa hàng đợi thêm và đặt proc như đặt proc vào hàng đợi sẵn sàng chiếm đoạt bằng cách so khớp ưu tiên. Thiết kế chính thuộc về chính sách MLQ được triển khai bởi get proc để tìm nạp một proc để phân phối tới CPU.

Mô tả về chính sách MLQ: bước duyệt qua danh sách hàng đợi sẵn sàng là một số được lập công thức cố định dựa trên mức độ ưu tiên, tức là vị trí: số tiến trình trên hàng đợi có mức độ ưu tiên = ưu tiên có thể chạy trước khi chuyển sang hàng đợi khác có mức độ ưu tiên khác (khe = MAX PRIO - trước).

Một ví dụ trong Linux MAX PRIO=140, prio=0..(MAX PRIO - 1)

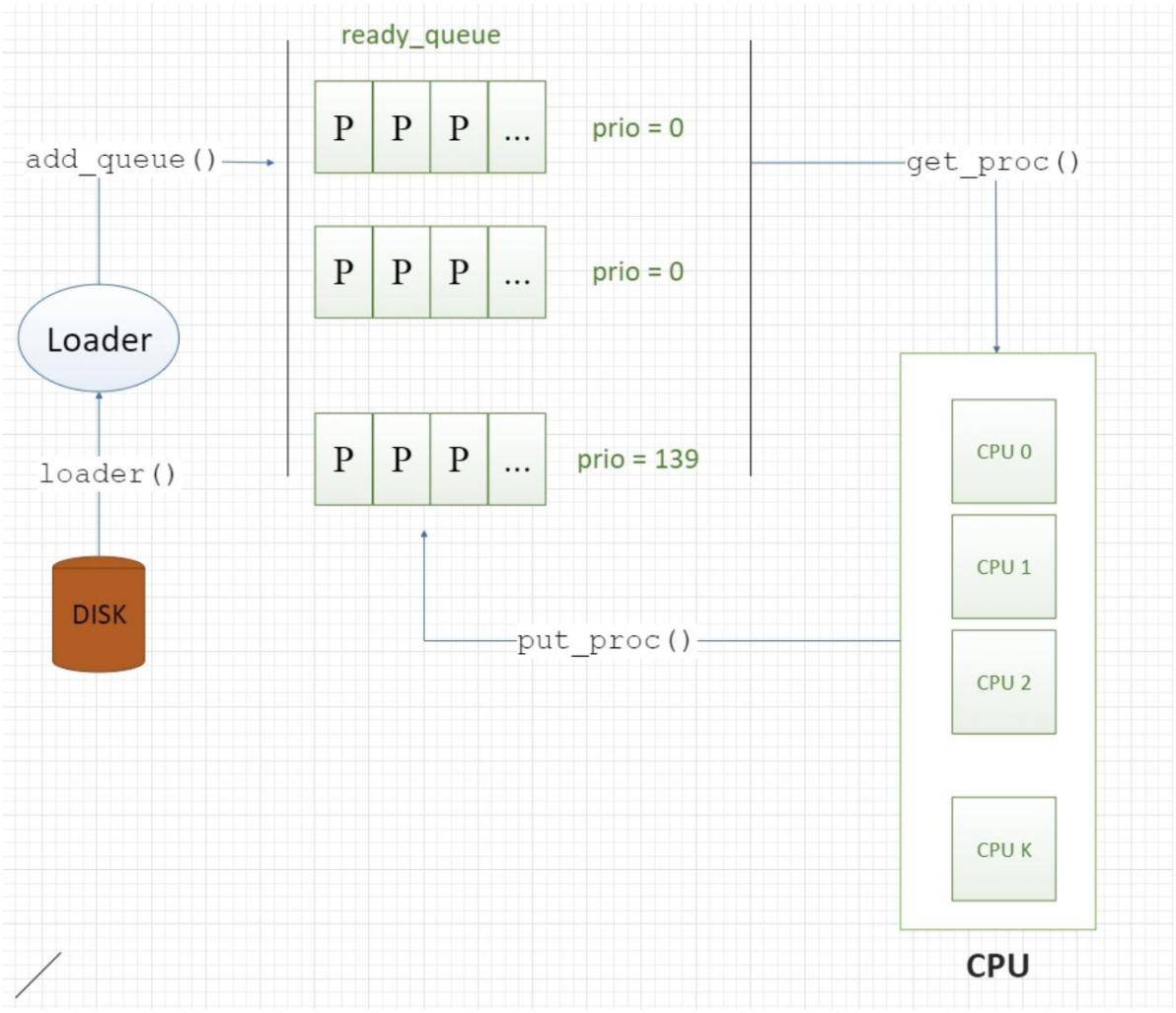
ưu tiên =	1	....	MAX_PRIO - 1	
0 vị trí = MAX_PRIO	MAX_PRIO - 1	....	1	

Giả sử chúng ta có MAX PRIO = 5 5 hàng đợi với mức độ ưu tiên tương ứng trong danh sách hàng đợi sẵn sàng. Hàng đợi đầu tiên có ưu tiên = 0 sẽ có vị trí = MAX PRIO - ưu tiên = 5 do đó quá trình trong hàng đợi này có thể được lên lịch để chạy 5 lần (nếu hàng đợi không trống), sau đó chuyển sang hàng đợi khác có mức độ ưu tiên thấp hơn. Chính sách MLQ chỉ đi qua bước cố định để duyệt qua tất cả hàng đợi trong danh sách hàng đợi sẵn sàng ưu tiên và quay lại hàng đợi bắt đầu.

Lưu ý: Bạn có thể triển khai chính sách MLQ của riêng mình miễn là chính sách này đảm bảo rằng tất cả các hàng đợi trong danh sách hàng đợi sẵn sàng phải lần lượt chạy các tác vụ của nó (không xuất hiện tình trạng chết đói) và hàng đợi có mức độ ưu tiên cao hơn có thể được lên lịch để chạy các tác vụ của nó một chút thường xuyên hơn so với hàng đợi ưu tiên thấp hơn.

Công việc của bạn trong phần này là thực hiện thuật toán này bằng cách hoàn thành các chức năng sau

- enqueue() và dequeue() (trong queue.c): Chúng ta đã định nghĩa một cấu trúc (queue t) cho hàng đợi ưu tiên tại queue.h. Nhiệm vụ của bạn là triển khai các chức năng đó để giúp đưa PCB mới vào hàng đợi và đưa PCB 'đến lượt' tiếp theo ra khỏi hàng đợi.
- get proc() (trong sched.c): lấy PCB của một tiến trình đang chờ từ hệ thống hàng đợi sẵn sàng. Các hàng đợi sẵn sàng được chọn 'lần lượt' đã được mô tả trong chính sách trên bằng MLQ.



Hình 2: Hoạt động của Scheduler trong giao việc

Để kiểm tra công việc của bạn, hãy biên dịch mã nguồn bằng Makefile

```
lên lịch trình
```

và sau đó chạy bộ lập lịch sử dụng cấu hình mẫu với

```
làm test_sched
```

Bạn có thể so sánh kết quả của mình với các câu trả lời mẫu trong thư mục đầu ra. Lưu ý rằng vì trình tải và bộ lập lịch chạy đồng thời và chính sách MLQ có thể khác do triển khai riêng lẻ nên có thể có nhiều hơn một câu trả lời đúng cho mỗi thử nghiệm.

Câu hỏi: Ưu điểm của việc sử dụng hàng đợi ưu tiên so với các thuật toán lập lịch khác mà bạn đã học là gì?

2.2 Quản lý bộ nhớ

Công cụ bộ nhớ ảo sử dụng cơ chế Phân đoạn với phân trang để quản lý bộ nhớ. Theo mặc định, kích thước của RAM ảo là 1 MB nên chúng ta phải sử dụng 20 bit để biểu thị địa chỉ của từng byte của nó. Với cơ chế phân đoạn có phân trang, chúng ta sử dụng 5 bit đầu tiên cho chỉ mục phân đoạn, 5 bit tiếp theo cho chỉ mục trang và 10 bit cuối cùng cho phần bù. Bạn có thể muốn xem lại ghi chú bài giảng chương 8 (trang #38 - #39) để hiểu cơ chế này hoạt động như thế nào. Sau đó, bạn có thể thực hiện quy trình dịch địa chỉ ảo của quy trình sang địa chỉ vật lý bằng cách hoàn thành các chức năng sau:

- get_trans_table() (trong mem.c): Tìm chỉ mục trang đã cho chỉ mục phân đoạn của quy trình.
- translate() (trong mem.c): sử dụng hàm get_trans_table() để dịch một địa chỉ ảo sang địa chỉ vật lý.

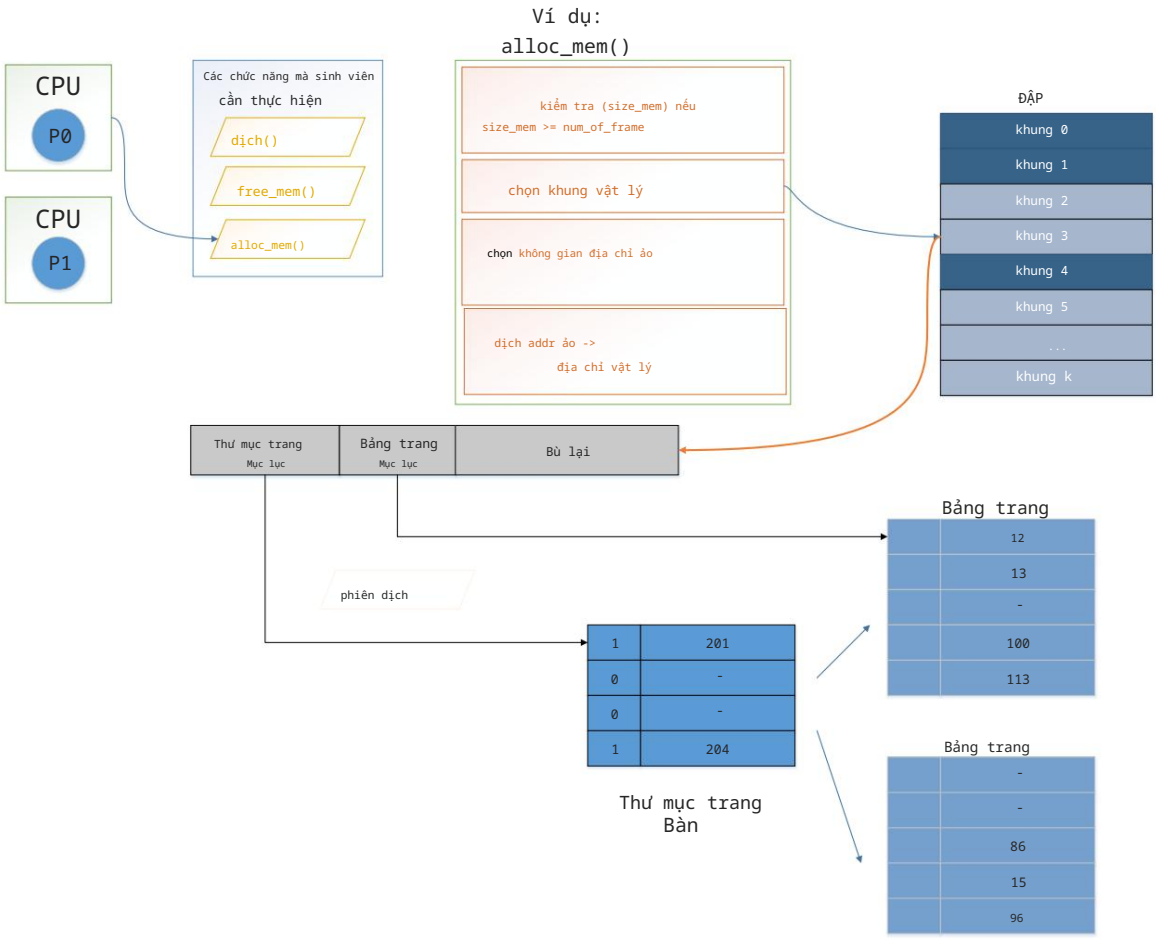
Lưu ý rằng các chức năng được đề cập ở trên được áp dụng cho các quy trình đã có bảng phân đoạn và bảng trang để chúng tôi thực hiện công việc của mình. Vấn đề chính là làm thế nào để xây dựng các bảng đó. Rõ ràng, chúng ta không thể ánh xạ toàn bộ không gian địa chỉ ảo của một tiến trình sang không gian địa chỉ của RAM vì RAM được chia sẻ bởi nhiều tiến trình và các vùng bộ nhớ được sử dụng bởi một tiến trình sẽ không được sử dụng bởi tiến trình khác (đảm bảo cách ly). Do đó, chúng ta nên bắt đầu quá trình với bảng phân đoạn trống và tiếp tục cập nhật nó mỗi khi quá trình cấp phát hoặc giải phóng bộ nhớ.

Để đơn giản hóa việc thực hiện phân bổ và giải phóng bộ nhớ, HĐH duy trì một cấu trúc đặc biệt gọi là mem_stat theo dõi trạng thái của các trang vật lý. Riêng RAM được chia thành nhiều trang theo cơ chế Segmentation with Paging và HĐH cấp phát bộ nhớ cho các tiến trình theo trang.
Nghĩa là, nếu quá trình yêu cầu dung lượng bộ nhớ nhỏ hơn kích thước trang, thì nó sẽ nhận được một trang hoàn toàn mới. Trách nhiệm của chỉ số mem là duy trì trạng thái (được sử dụng/miễn phí) của các trang đó. Riêng mem_stat là một bảng trong đó mỗi hàng là một cấu trúc được định nghĩa như sau

```
cấu trúc {
        uint32_t proc; chỉ
        mục uint32_t; uint32_t
        tiếp theo;
}
```

trong đó proc là PID của quy trình hiện đang sử dụng trang này. Nếu proc = 0, trang này miễn phí và hệ điều hành có thể phân bổ nó cho bất kỳ quy trình nào. Quá trình có thể yêu cầu một vùng bộ nhớ có kích thước lớn hơn nhiều so với kích thước của một trang. Trong trường hợp này, HĐH phải phân bổ nhiều trang cho quy trình này, mỗi chỉ số mem phải có cùng giá trị proc. Vì quá trình sử dụng địa chỉ ảo để truy cập nội dung của RAM nên chúng ta phải để địa chỉ ảo của các trang được cấp phát liền kề nhau nhưng không cần phải có địa chỉ vật lý của chúng.
Ví dụ: hỗ trợ một quy trình yêu cầu 2 trang và chúng tôi có chính xác 2 trang trống trong RAM nhưng các trang đó không liền kề nhau: địa chỉ của byte đầu tiên của chúng lần lượt là 0x00000 và 0x01000 (chúng tôi biểu thị bộ nhớ 20 bit dưới dạng số thập lục phân 5). Trong không gian địa chỉ ảo, chúng tôi tạo hai trang liền kề có byte đầu tiên là 0x00000 và 0x00400 (độ lệch kéo dài 10 bit) và ánh xạ trang 0x00000 sang trang vật lý 0x00000, trang 0x00400 sang trang vật lý 0x01000. Mặc dù tiến trình cảm thấy rằng các vùng bộ nhớ mà nó đã cấp phát là liền kề nhau, nhưng thực ra không phải vậy. Vì chúng tôi sử dụng phương pháp này để cấp phát bộ nhớ nên một hàng trong chỉ số mem có trường chỉ mục để biết cho chúng tôi biết vị trí của nó trong danh sách trang được cấp phát. Trường tiếp theo được sử dụng để chỉ ra chỉ mục trong chỉ số mem của trang tiếp theo trong danh sách các trang được phân bổ. Nếu trang là trang cuối cùng, trường này được đặt thành -1. Trong ví dụ trước, trang 0x00000 có chỉ mục = 0 và trang 0x01000 sẽ là 1 trong khi giá trị của trang tiếp theo lần lượt là 4 và -1 (vì chúng tôi đã loại bỏ phần bù 10 bit trong địa chỉ vật lý ban đầu).

Hình 3 cho thấy cách chúng ta phân bổ các vùng bộ nhớ mới và tạo mục nhập mới trong phân đoạn và bảng trang bên trong một quy trình. Đặc biệt, đối với mỗi trang mới được phân bổ, chúng ta phải thêm mục mới vào bảng trang theo số phân đoạn và số trang của trang này.



Hình 3: Thao tác liên quan đến bộ nhớ ảo trong bài tập

Nhiệm vụ chính của bạn trong phần này là triển khai các hàm alloc mem() và free mem(), cả hai đều được xác định trong mem.c dựa trên thuật toán được mô tả ở trên.

Để xác minh việc triển khai của bạn, trước tiên bạn phải biên dịch mã bằng Makefile:

```
làm mem
```

và sau đó chạy thử nghiệm trên cấu hình mẫu

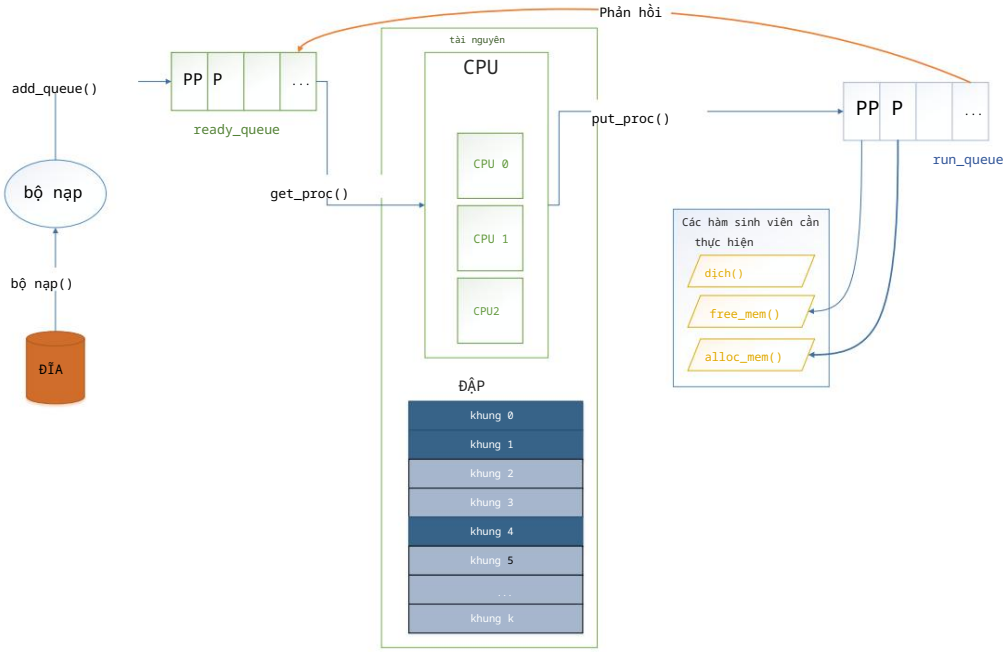
```
làm test_mem
```

Bạn có thể so sánh kết quả của mình với câu trả lời mẫu trong thư mục đầu ra. Kết quả của bạn phải giống với câu trả lời.

Câu hỏi Ưu điểm và nhược điểm của phân đoạn bằng phân trang là gì.

2.3 Kết hợp tất cả lại với nhau

Cuối cùng, chúng tôi kết hợp bộ lập lịch và Công cụ bộ nhớ ảo để tạo thành một hệ điều hành hoàn chỉnh. Hình 4 cho thấy tổ chức hoàn chỉnh của HĐH. Nhiệm vụ cuối cùng cần làm là đồng bộ hóa. Vì hệ điều hành chạy trên nhiều bộ xử lý, nên có thể nhiều tài nguyên chia sẻ có thể được truy cập đồng thời bởi nhiều quy trình cùng một lúc. Công việc của bạn trong phần này là tìm tài nguyên chia sẻ và sử dụng cơ chế khóa để bảo vệ chúng.



Hình 4: Mô hình phân công 2

Kiểm tra công việc của bạn bằng cách biên dịch toàn bộ mã nguồn

```
làm cho tất cả
```

và sau đó kiểm tra kết quả của bạn với các câu trả lời mẫu bằng cách chạy lệnh sau

```
làm test_all
```

và so sánh đầu ra của bạn với đầu ra. Hãy nhớ rằng khi chúng ta đang chạy nhiều quy trình, có thể có nhiều hơn một kết quả đúng.

Câu hỏi: Điều gì sẽ xảy ra nếu việc đồng bộ hóa không được xử lý trong HĐH đơn giản của bạn? Minh họa bằng ví dụ vấn đề của hệ điều hành đơn giản của bạn nếu bạn có.

3 đệ trình

3.1 Mã nguồn

Yêu cầu: bạn phải viết mã lệnh gọi hệ thống theo kiểu viết mã. Tham khảo: https://www.gnu.org/prep/standards/html node/Writing-C.html.

3.2 Báo cáo

Viết một báo cáo ngắn trả lời các câu hỏi trong phần triển khai và giải thích kết quả chạy thử nghiệm trong mỗi phần:

- Lập lịch: vẽ sơ đồ Gantt mô tả các tiến trình được thực hiện bởi CPU như thế nào.
- Bộ nhớ: Hiển thị trạng thái của RAM sau mỗi lần gọi chức năng cấp phát và hủy cấp phát bộ nhớ.
- Nhìn chung: sinh viên tự tìm cách diễn giải kết quả mô phỏng.

Sau khi bạn hoàn thành bài tập, hãy chuyển báo cáo của bạn vào thư mục mã nguồn và nén toàn bộ thư mục thành một tệp duy nhất có tên bài tập MSSV.zip và gửi lên trang Elearning.

3.3 Chấm điểm

Bạn phải thực hiện nhiệm vụ này theo nhóm hai hoặc ba người. Điểm chung cho mỗi học sinh là sự kết hợp của hai phần

- Nhóm: nhóm của bạn thực hiện nhiệm vụ tốt như thế nào (6 điểm)
  - Lập kế hoạch (1,5 điểm)
  - Trí nhớ (1,5 điểm)
  - Đồng bộ hóa (1,5 điểm)
  - Báo cáo (1,5 điểm)
- Cá nhân: bạn đóng góp bao nhiêu cho nhóm và bạn hiểu bài tập đến đâu (4 điểm). Bạn nên đánh dấu tất cả đóng góp của mình vào bản báo cáo.