ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

HỆ ĐIỀU HÀNH (CO2018)

Simple Operating System

Giáo viên hướng dẫn: Nguyễn Thanh Quân

Sinh viên hiện thực:	Đoàn Thị Huế	2113447
	Nguyễn Anh Tấn	2114741
	Huỳnh Hữu Tín	2110584

Tp. Hồ Chí Minh, 12/2023

Bảng phân chia công việc

STT	Họ và tên	MSSV	Nhiệm vụ	Mức độ hoàn thành
1	Nguyễn Anh Tấn	2114741	Scheduler	100%
2	Đoàn Thị Huế	2113447	Memory	100%
3	Huỳnh Hữu Tín	2110584	Memory	100%

Contents

1 Scheduler(Lập lịch)

1.1 Trả lời câu hỏi

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

Trả lời:

Một số thuật toán lập lịch :

First Come First Served (FCFS): Là giải thuật đơn giản nhất, giống với giải thuật FIFO, tiến trình nào yêu cầu CPU trước sẽ được phân phối CPU trước.

Hạn chế của thuật toán

- Do sử dụng chiến lược không ưu tiên, nếu một tiến trình bắt đầu, CPU sẽ thực thi tiến trình đó cho đến khi nó kết thúc.
- Nếu một tiến trình có chu kỳ CPU dài thì các tiến trình phía sau (có thể là các tiến trình có chu kỳ CPU ngắn) trong hàng đợi phải chờ rất lâu

Shortest Job First (SJF): Gắn với mỗi tiến trình là thời gian sử dụng CPU tiếp sau của nó, thời giân này được sử dụng để lập lịch các tiến trình với hời gian đợi là ngắn nhất.

Hạn chế của thuật toán:

- Cần phải ước tính trước thời gian CPU tiếp theo của tiến trình khi điều này thường không thể thực hiện được trong thực tế.
- Các quy trình có thời lượng CPU dài sẽ có nhiều thời gian chờ đợi hơn hoặc bị trì hoãn vô thời hạn khi nhiều quy trình có thời lượng CPU ngắn đồng thời vào hàng đợi -> stavation

Round Robin (RR): Thực hiện theo nguyên tắc quay vòng, lần lượt mỗi tác vụ sẽ được chạy trong một hàng đợi với giới hạn thời gian cụ thể(time quantum). Khả năng thực thi của thuật toán liên tục, tuần hoàn.

Hạn chế của thuật toán:

- Nếu quantum quá lớn: RR->FCFS.
- Nếu quantum quá nhỏ thì khả năng chuyển ngữ cảnh của CPU sẽ tăng lên tốn nhiều chi phí và thời gian.

Priority Scheduling: Tại thời điểm xuất hiện một quá trình trong hàng đợi sẵn sàng, Mức độ ưu tiên của nó được so sánh với mức độ ưu tiên của các quá trình khác có trong hàng đợi sẵn sàng cũng như với một quá trình đang được CPU thực thi tại thời điểm đó của thời gian. Cái có mức độ ưu tiên cao nhất trong số tất cả các quy trình có sẵn sẽ được cấp cho CPU tiếp theo.

Hạn chế của thuật toán:

- Các tiến trình có mức độ ưu tiên thấp hơn có thể không có cơ hội thực thi -> stavation.

Trong bài tập để tận dụng các ưu điểm và hạn chế các khuyết điểm ta có thực hiện giải thuật MLQ(Multilevel Queue), trong đó, mỗi hàng đợi ứng với một priority. Các hàng có priority cao sẽ được thực thi trước, các process trong một hàng đợi được thực thi theo giải thuật Round-Robin. Mỗi queue sẽ có một số slot cố định, khi các slot này được dùng hết thì hệ thống sẽ chuyển sang queue có độ ưu tiên thấp hơn.

Ưu điểm của **Multi-Level Queue (MLQ):**

- **Chi phí lập lịch thấp:** Do các quy trình được gán vĩnh viễn cho các hàng đợi tương ứng nên chi phí lập lịch thấp, vì người lập lịch chỉ cần chọn hàng đợi thích hợp để thực thi.
- **Phân bổ thời gian CPU hiệu quả:** Thuật toán lập lịch đảm bảo rằng các quy trình có mức ưu tiên cao hơn được thực thi kịp thời, trong khi vẫn cho phép các quy trình có mức ưu tiên thấp hơn thực thi khi CPU không hoạt động. Điều này đảm bảo sử dụng tối ưu thời gian CPU.
- **Tính công bằng:** Thuật toán lập lịch cung cấp sự phân bổ hợp lý thời gian CPU cho các loại quy trình khác nhau, dựa trên mức độ ưu tiên và yêu cầu của chúng.
- **Có thể tùy chỉnh:** Thuật toán lập lịch có thể được tùy chỉnh để đáp ứng các yêu cầu cụ thể của các loại quy trình khác nhau. Các thuật toán lập lịch khác nhau có thể được sử dụng cho mỗi hàng đợi, tùy thuộc vào yêu cầu của các tiến trình trong hàng đợi đó.
- **Ưu tiên:** Mức độ ưu tiên được chỉ định cho các quy trình dựa trên đặc điểm và tầm quan trọng của chúng, điều này đảm bảo rằng các quy trình quan trọng được thực hiện kịp thời.
- **Quyền ưu tiên:** Quyền ưu tiên được cho phép trong MLQ, có nghĩa là các quy trình có mức độ ưu tiên cao hơn có thể giành quyền ưu tiên các quy trình có mức độ ưu tiên thấp hơn và CPU được phân bổ cho quy trình có mức độ ưu tiên cao hơn. Điều này giúp đảm bảo rằng các quy trình có mức độ ưu tiên cao được thực hiện kịp thời.

1.2 Hiện thực chương trình

1.2.1 Module 1

- Input được lấy từ file my_sched_0.

```
3 1 6
1048576 16777216 0 0 0
0 sched_proc_1 138
1 sched_proc_2 138
2 sched_proc_3 135
3 sched_proc_4 134
4 sched_proc_5 139
5 sched_proc_6 139
```

- Output là file my_sched_0 trong file output.

```
Time slot   0
ld_routine
        Loaded a process at input/proc/sched_proc_1, PID: 1
        PRIO: 138
Time slot   1
        CPU 0: Dispatched process  1
        Loaded a process at input/proc/sched_proc_2, PID: 2
        PRIO: 138
Time slot   2
        Loaded a process at input/proc/sched_proc_3, PID: 3
        PRIO: 135
Time slot   3
        Loaded a process at input/proc/sched_proc_4, PID: 4
        PRIO: 134
Time slot   4
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
        Loaded a process at input/proc/sched_proc_5, PID: 5
        PRIO: 139
Time slot   5
        Loaded a process at input/proc/sched_proc_6, PID: 6
        PRIO: 139
Time slot   6
Time slot   7
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot   8
Time slot   9
Time slot  10
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot  11
Time slot  12
Time slot  13
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot  14
Time slot  15
Time slot  16
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot  17
Time slot  18
Time slot  19
        CPU 0: Processed  4 has finished
        CPU 0: Dispatched process  3
Time slot  20
Time slot  21
Time slot  22
        CPU 0: Put process  3 to run queue
```

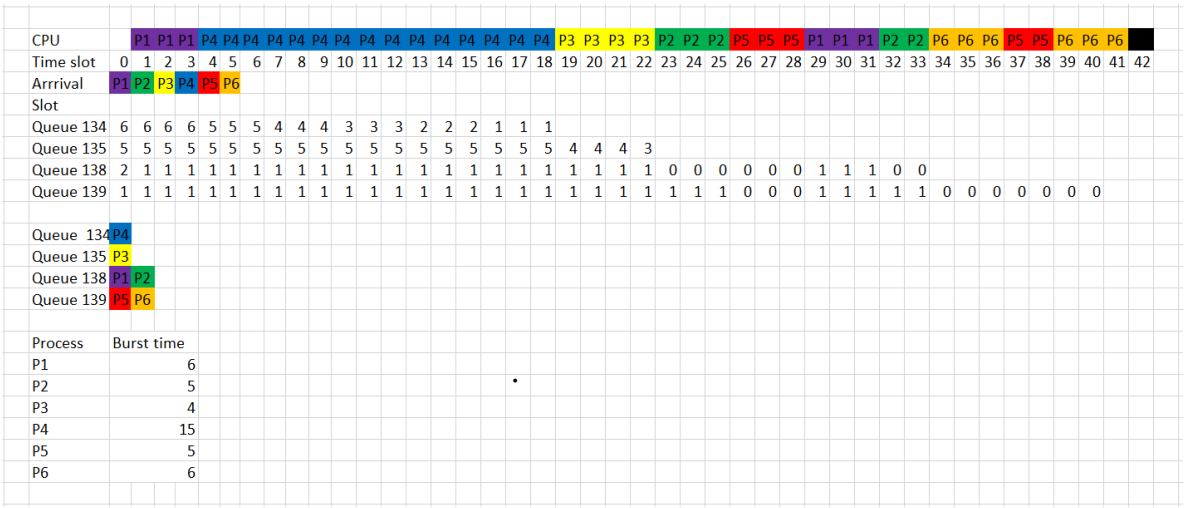
```
        CPU 0: Dispatched process  3
Time slot  23
        CPU 0: Processed  3 has finished
        CPU 0: Dispatched process  2
Time slot  24
Time slot  25
Time slot  26
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  5
Time slot  27
Time slot  28
Time slot  29
        CPU 0: Put process  5 to run queue
        CPU 0: Dispatched process  1
Time slot  30
Time slot  31
Time slot  32
        CPU 0: Processed  1 has finished
        CPU 0: Dispatched process  2
Time slot  33
Time slot  34
        CPU 0: Processed  2 has finished
        CPU 0: Dispatched process  6
Time slot  35
Time slot  36
Time slot  37
        CPU 0: Put process  6 to run queue
        CPU 0: Dispatched process  5
Time slot  38
Time slot  39
        CPU 0: Processed  5 has finished
        CPU 0: Dispatched process  6
Time slot  40
Time slot  41
Time slot  42
        CPU 0: Processed  6 has finished
        CPU 0 stopped

```

- File đầu vào chứa các thông tin quan trọng sau:
  - Time slice: 3
  - Số lượng CPU: 1
  - Số lượng tiến trình: 6
- Bảng mô tả các tiến trình

Arrival time	Process name	Live priority	Burst time
0	sched_proc_1	138	6
1	sched_proc_2	138	5
2	sched_proc_3	135	4
3	sched_proc_4	134	15
4	sched_proc_5	139	5
5	sched_proc_6	139	6

- Bảng mô phỏng quá trình thực thi



Hình 1: Bảng mô phỏng quá trình thực thi Module 1

- Giải thích quá trình thực thi
  - Tại time slot 1: p1 đã được load vào queue 138 (tại đầu time slot 0) và queue vẫn còn slot nên p1 được cấp CPU để chạy. Queue 138 còn 1 slot.

  - Tại time slot 4: Lần lượt p2, p3, p4 được nạp vào các queue. Lúc này hết time slice (3) nên p1 bị preemptived, lúc này p4 có priority cao nhất và queue 134 còn slot nên p4 được cấp CPU để chạy.

  - Tại time slot 7, 10, 13, 16: Lần lượt p5, p6 được load tại time slot 4, 5. Tại các time slot này thì time slice hết, p4 bị preemptived nhưng queue 134 vẫn còn slot và p4 vẫn có priority cao nhất nên p4 tiếp tục được cấp phát CPU. Tiến trình p4 hoàn thành sau khi hết time slot 18. Queue 134 còn 1 (6-5) slot.

  - Tại time slot 19: Lúc này tiến trình có độ ưu tiên cao nhất là p3 và p3 nằm trong Queue có priority cao nhất có tiến trình và còn slot nên p3 được cấp phát CPU để chạy.

  - Tại time slot 22: Tiến trình p3 bị preemptive vì hết time slice. Tuy nhiên queue 135 vẫn còn slot (4) nên p3 tiếp tục được cấp phát CPU. Vì thời gian chạy của p3 chỉ còn 1 đơn vị thời gian nên sau khi hết time slot 22 thì p3 hoàn thành.

- Tại time slot 23: p2 là tiến trình nằm trong Queue 138 có priority cao nhất hiện tại và còn slot (1) nên p2 được cấp phát tài nguyên. Queue 138 chính thức hết slot.

- Tại time slot 26: Tiến trình p2 bị preemptived. Vì queue 138 hết slot nên p2 không thể được cấp CPU lại được nữa, thay vào đó, CPU sẽ được cấp cho tiến trình nằm trong queue có độ ưu tiên thấp hơn là p5 và p6. Tiến trình p5 được chọn để cấp phát CPU. Queue 139 hết slot

- Tại time slot 29: Tiến trình p5 bị preemptived. Vì queue 139 hết slot nên p5 không thể được chọn lại nữa. Vì queue 139 là queue cuối cùng nên OS tiến hành reset lại slot cho toàn bộ queue. Lúc này queue có priority cao nhất có tiến trình là queue 138 với tiến trình là p1 nên p1 được cấp phát CPU. Tiến trình p1 hoàn thành tại cuối time slot 31. Queue 138 còn 1 slot.

- Tại time slot 32: Hết time slice, p1 bị preemptived. Queue có priority cao nhất có tiến trình là queue 138 và còn 1 slot nên tiến trình p2 được cấp phát CPU. Tiến trình p2 chạy được 2 time slot thì hoàn thành vào cuối time slot 33.

- Tại time slot 34. 37.39: Chỉ còn queue 139 là còn tiến trình và queue 139 chỉ có 1 slot nên p5, p6 lần lượt được cấp phát CPU xen kẽ sau mỗi lần tiến trình còn lại bị preemptived. Tiến trình p5 hoàn thành vào cuối time slot 38 còn p6 hoàn thành vào cuối time slot 41. CPU dừng lại tại time slot 42.

1.2.2 Module 2

- Input được lấy từ file my_sched_1.

```
3 3 6
1048576 16777216 0 0 0
0 sched_proc_1 138
1 sched_proc_2 138
2 sched_proc_3 135
3 sched_proc_4 134
4 sched_proc_5 139
5 sched_proc_6 139
```

- Output là file my_sched_1 trong file output.

```
Time slot   0
ld_routine
        Loaded a process at input/proc/sched_proc_1, PID: 1
        PRIO: 138
Time slot   1
        CPU 2: Dispatched process  1
        Loaded a process at input/proc/sched_proc_2, PID: 2
        PRIO: 138
Time slot   2
        CPU 1: Dispatched process  2
```

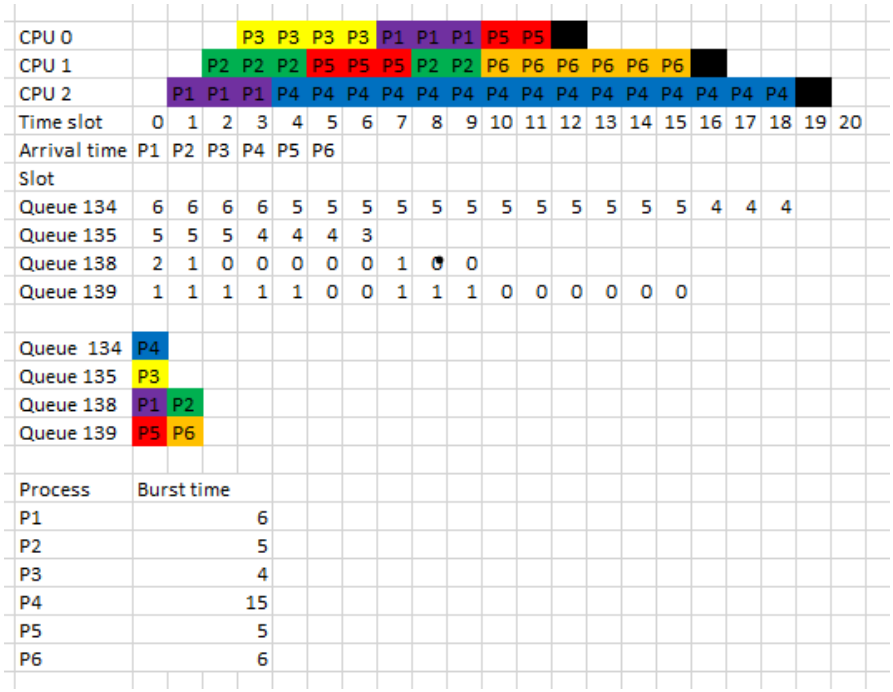
```
        Loaded a process at input/proc/sched_proc_3, PID: 3
        PRIO: 135
Time slot   3
        CPU 0: Dispatched process  3
        Loaded a process at input/proc/sched_proc_4, PID: 4
        PRIO: 134
Time slot   4
        CPU 2: Put process  1 to run queue
        CPU 2: Dispatched process  4
        Loaded a process at input/proc/sched_proc_5, PID: 5
        PRIO: 139
Time slot   5
        CPU 1: Put process  2 to run queue
        CPU 1: Dispatched process  5
        Loaded a process at input/proc/sched_proc_6, PID: 6
        PRIO: 139
Time slot   6
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  3
Time slot   7
        CPU 2: Put process  4 to run queue
        CPU 2: Dispatched process  4
        CPU 0: Processed  3 has finished
        CPU 0: Dispatched process  1
Time slot   8
        CPU 1: Put process  5 to run queue
        CPU 1: Dispatched process  2
Time slot   9
Time slot  10
        CPU 2: Put process  4 to run queue
        CPU 2: Dispatched process  4
        CPU 1: Processed  2 has finished
        CPU 0: Processed  1 has finished
        CPU 1: Dispatched process  6
        CPU 0: Dispatched process  5
Time slot  11
Time slot  12
        CPU 0: Processed  5 has finished
        CPU 0 stopped
Time slot  13
        CPU 2: Put process  4 to run queue
        CPU 2: Dispatched process  4
        CPU 1: Put process  6 to run queue
        CPU 1: Dispatched process  6
Time slot  14
Time slot  15
Time slot  16
        CPU 2: Put process  4 to run queue
        CPU 2: Dispatched process  4
        CPU 1: Processed  6 has finished
```

```
55          CPU 1 stopped
56  Time slot   17
57  Time slot   18
58  Time slot   19
59          CPU 2: Processed   4 has finished
60          CPU 2 stopped
```

- File đầu vào chứa các thông tin quan trọng sau:
  - Time slice: 3
  - Số lượng CPU: 3
  - Số lượng tiến trình: 6
- Bảng mô tả các tiến trình

Arrival time	Process name	Live priority	Burst time
0	sched_proc_1	138	6
1	sched_proc_2	138	5
2	sched_proc_3	135	4
3	sched_proc_4	134	15
4	sched_proc_5	139	5
5	sched_proc_6	139	6

- Bảng mô phỏng quá trình thực thi



Hình 2: Bảng mô phỏng quá trình thực thi Module 2

- Giải thích quá trình thực thi
  - Tại time slot 1, 2, 3: Tiến trình p1, p2, p3 lần lượt được chọn để cấp phát CPU. Queue 138 lúc này còn lại 0 slot.
  - Tại time slot 4: Tiến trình p4 đã được tải vào queue 134. Tiến trình p1 vừa được preemptived và p4 thì nằm trong queue có priority cao hơn nên p4 được cấp phát CPU.
  - Tại time slot 5: Tiến trình p5 đã được tải vào queue 139. Tiến trình p2 vừa được preemptived và queue 138 hết slot nên p5 được cấp phát CPU. Vì vừa cấp phát CPU cho p5 nên queue 139 hết slot, lúc này OS tiến hành cấp phát lại toàn bộ slot cho các queue.
  - Tại time slot 7: p3 và p4 lần lượt preemptived, nhưng vì queue 138 đã được cấp slot lại và lúc này có 2 tiến trình cần cấp phát CPU nên p1 và p4 được chọn. Queue 138 còn 1 slot. Tiến trình p3 hoàn thành tại cuối time slot 5.
  - Tại time slot 10: Các tiến trình p1, p2 lần lượt hoàn thành, còn p4 thì bị preemptived. Queue 134 còn slot và có priority cao nhất nên p4 tiếp tục được cấp phát CPU. Các tiến trình p5, p6 lần lượt được cấp phát CPU 1, 2 trong quá trình cấp phát CPU cho 2 tiến trình này, diễn ra quá trình cấp phát lại toàn bộ slot cho các queue.
  - Sau time slot 10: Các tiến trình p5, p6, p4 lần lượt hoàn thành và bị preemptived. Tiến trình p5 hoàn thành vào cuối time slot 11, p6 hoàn thành vào cuối time slot 15 còn p4 hoàn thành vào cuối time slot 18. Các CPU 1, 2, 0 lần lượt dừng lại theo thứ tự kết thúc của p5, p6, p4.

2 Memory management(Quản lý bộ nhớ)

2.1 Trả lời câu hỏi

2.1.1 Ánh xạ bộ nhớ ảo trong mỗi tiến trình

Câu hỏi: Trong hệ điều hành đơn giản này, chúng ta triển khai một thiết kế với nhiều phân đoạn nhớ hoặc khu vực nhớ trong khai báo mã nguồn. Lợi ích của thiết kế đa đoạn được đề xuất là gì?

Trả lời:

Trong hệ điều hành, phân đoạn là một kỹ thuật quản lý bộ nhớ trong đó bộ nhớ được chia thành các phần có kích thước thay đổi được. Mỗi phần được gọi là một phân đoạn có thể được phân bổ cho một quá trình.
Ưu điểm của phân đoạn:

- Tăng cường bảo mật và an toàn: Mỗi đoạn có thể được bảo vệ riêng biệt, ngăn chặn việc truy cập không hợp lệ từ các đoạn khác. Điều này giúp ngăn chặn các lỗi và các cuộc tấn công bảo mật.
- Tăng hiệu suất và giảm độ phức tạp: Việc chia nhỏ bộ nhớ thành các đoạn nhỏ hơn giúp giảm bớt độ phức tạp của việc quản lý bộ nhớ. Đồng thời, việc chia nhỏ bộ nhớ cũng giúp tăng hiệu suất của hệ thống, bởi vì mỗi đoạn có thể được quản lý và xử lý độc lập.
- Tăng khả năng quản lý bộ nhớ: Thiết kế đa đoạn cho phép hệ thống quản lý bộ nhớ một cách linh hoạt hơn, vì mỗi đoạn có thể được cấp phát, thay đổi kích thước hoặc di chuyển trong bộ nhớ mà không ảnh hưởng đến các đoạn khác.
- Cải thiện hiệu suất của bộ nhớ đệm: Thiết kế đa đoạn giúp giảm số lượng không gian nhớ không sử dụng (fragmentation) bằng cách cho phép các đoạn có kích thước khác nhau. Điều này giúp tăng hiệu suất của bộ nhớ đệm và giảm thiểu việc mất mát bộ nhớ.

2.1.2 Bộ nhớ vật lý của hệ thống

Câu hỏi: Điều gì sẽ xảy ra nếu chúng ta chia địa chỉ nhiều hơn 2 cấp trong hệ thống quản lý bộ nhớ phân trang?

Trả lời:

Nếu chúng ta chia địa chỉ nhiều hơn 2 cấp trong hệ thống quản lý bộ nhớ phân trang, chúng ta sẽ sử dụng một cấu trúc gọi là phân trang đa cấp (Multilevel Paging). Đây là một cách tiếp cận khác để giảm bớt lượng bộ nhớ cần thiết cho bảng định tuyến trang (page table), đặc biệt là trong các hệ thống có không gian địa chỉ lớn.

Phân trang đa cấp hoạt động bằng cách chia bảng định tuyến trang thành nhiều cấp, mỗi cấp là một bảng định tuyến trang nhỏ hơn. Các cấp trên sẽ chứa các con trỏ đến các bảng định tuyến trang cấp dưới. Cấp cuối cùng sẽ chứa thông tin thực tế về khung (frame).
Các lợi ích của phân trang đa cấp bao gồm:

- Giảm bớt lượng bộ nhớ cần thiết cho bảng định tuyến trang: Mỗi cấp trong phân trang đa cấp chứa ít hơn số mục so với một bảng định tuyến trang đơn lẻ, do đó cần ít bộ nhớ hơn. Điều này giúp giảm bớt lượng bộ nhớ cần thiết cho bảng định tuyến trang.

- Tăng tốc độ tìm kiếm bảng định tuyến trang: Với số lượng mục nhỏ hơn trên mỗi cấp, việc tìm kiếm bảng định tuyến trang sẽ nhanh hơn, dẫn đến hiệu suất hệ thống tổng thể tốt hơn.
- Linh hoạt hơn trong việc sắp xếp không gian bộ nhớ: Phân trang đa cấp cung cấp linh hoạt hơn trong việc sắp xếp không gian bộ nhớ. Điều này đặc biệt hữu ích trong các hệ thống có yêu cầu bộ nhớ thay đổi, vì nó cho phép bảng định tuyến trang được điều chỉnh để phù hợp với nhu cầu thay đổi.

2.1.3 Phương pháp dịch địa chỉ dựa trên phân trang

Câu hỏi: Ưu điểm và nhược điểm của việc phân đoạn kết hợp với phân trang là gì?

Trả lời:

Phân đoạn kết hợp với phân trang là một phương pháp quản lý bộ nhớ hiệu quả, nhưng cũng có một số ưu và nhược điểm mà chúng ta cần chú ý đến.
Về mặt ưu điểm:

- Hiệu quả về bộ nhớ: Phân trang giúp loại bỏ sự phân mảnh bên ngoài, đảm bảo sử dụng hiệu quả bộ nhớ chính. Các phần di chuyển vào và ra khỏi bộ nhớ chính được cố định và có cùng kích thước, do đó có thể phát triển các thuật toán quản lý bộ nhớ tinh vi khai thác hành vi của chương trình.
- Quản lý bộ nhớ hợp lý: Phân đoạn cho phép phân vùng hợp lý và bảo vệ các thành phần ứng dụng, trong khi phân trang thì không.
- Hỗ trợ trao đổi và bảo vệ: Phân đoạn được hiển thị cho nhà phát triển và có khả năng quản lý sự tăng trưởng của cấu trúc dữ liệu, mô đun hóa và hỗ trợ trao đổi và bảo vệ.

Về mặt nhược điểm:

- Khó khăn trong quản lý bộ nhớ: Phân đoạn có nhiều không gian địa chỉ tuyến tính và phân trang chỉ có một. Điều này có thể gây ra sự khó khăn trong việc quản lý bộ nhớ.
- Thời gian: Có sự chậm trễ khi chúng ta truy cập vào bộ nhớ. Ngoài ra, phần cứng cần thiết để thực hiện kỹ thuật phân trang theo trang rất phức tạp.

2.1.4 Tổng hợp

Câu hỏi: Nếu không xử lý đồng bộ hóa trong hệ điều hành đơn giản của bạn, sẽ xảy ra vấn đề gì cho ví dụ minh họa nếu có?

Trả lời:

Nếu việc đồng bộ hóa không được xử lý trong hệ điều hành, nó có thể xảy ra nhiều vấn đề, một trong đó là việc dữ liệu không nhất quán.

Cùng xét ví dụ sau để hiểu sâu hơn về vấn đề này:

Quy trình A và Quy trình B là hai quy trình đồng thời tăng và giảm biến count được chia sẻ. Cả hai quá trình đều bắt đầu với số đếm được khởi tạo bằng 0.

- Quá trình A thực thi đoạn mã sau: count = count + 1

- Quá trình B thực thi đoạn mã sau: count = count 1

Nếu không đồng bộ hóa, chuỗi sự kiện sau có thể xảy ra:

- Tiến trình A đọc giá trị hiện tại của số đếm (0) vào thanh ghi.
- Quá trình B đọc giá trị hiện tại của số đếm (0) vào thanh ghi.
- Tiến trình A tăng giá trị thanh ghi cục bộ của nó lên 1 (0 + 1 = 1).
- Quá trình B giảm giá trị thanh ghi cục bộ của nó đi 1 (0 - 1 = -1).
- Quá trình A ghi lại giá trị cập nhật của nó (1) để đếm.
- Quá trình B ghi lại giá trị cập nhật của nó (-1) để đếm.

Trong kịch bản này, cả hai tiến trình đã thực thi các hoạt động của chúng đồng thời mà không có bất kỳ đồng bộ hóa. Kết quả là giá trị cuối cùng của count không chính xác (-1 thay vì 0).
Vấn đề nàyđược gọi là "race condition", trong đó đầu ra của chương trình phụ thuộc vào định thời và xen kẽ các hoạt động đồng thời.

Ngoài ra, nếu không có cơ chế đồng bộ phù hợp sẽ có nguy cơ bị "deadlock". Deadlock có thể xảy ra khi hai hoặc nhiều tiến trình đang chờ vô thời hạn để giải phóng tài nguyên. Nếu có các tài nguyên được chia sẻ mà các tiến trình cần có quyền truy cập độc quyền mà không cần đồng bộ hóa thích hợp, có thể xảy ra trường hợp có nhiều quy trình chờ đợi một tài nguyên sẽ không bao giờ được giải phóng.

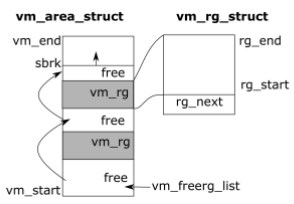
2.2 Hiện thực memory

2.2.1 Cấu trúc bộ nhớ

Trong hệ điều hành này, bộ nhớ được thiết kế gồm các phần sau:

- Mỗi process gồm một memory mapping để quản lý các vùng nhớ của process đó và các con trỏ chỉ tới RAM và active-SWAP mà process sử dụng.
- Memory mapping là một cấu trúc gồm danh sách các memory area, các memory region và một list fifo-page gồm các page trống của process. Ngoài ra, memory mapping cung cấp một danh sách các page entry để map giữa các page của process và frame tương ứng trong bộ nhớ.
- Memory area tương ứng với các segment của một process và có kích thước thay đổi. Mỗi memory area bao gồm nhiều memory region cũng có kích thước không cố định.
- Trong hệ thống này, ta sẽ dụng một bộ nhớ chính là RAM và bộ nhớ thứ cấp là SWAP (ta được hỗ trợ tối đa 4 SWAP). Các bộ nhớ có chung một kiểu cấu tạo, gồm 2 danh sách lần lượt chứa các frame trống và các frame đang được sử dụng.

Cấu trúc của Memory Area và Memory region:



2.2.2 Trạng thái RAM sau khi được cấp phát và thu hồi

- Input: Ta xét file `os_1_singleCPU_mlq`

```
2 1  8
1 s4   4
2 s3   3
4 m1s  2
6 s2   3
7 m0s  3
9 p1s  2
11 s0  1
16 s1  0
```

- Output: `os_1_singleCPU_mlq.output`

```
Time slot   0
ld_routine
Time slot   1
        Loaded a process at input/proc/s4, PID: 1 PRIO: 4
Time slot   2
        Loaded a process at input/proc/s3, PID: 2 PRIO: 3
Time slot   3
        CPU 0: Dispatched process  1
Time slot   4
        Loaded a process at input/proc/m1s, PID: 3 PRIO: 2
Time slot   5
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  2
Time slot   6
        Loaded a process at input/proc/s2, PID: 4 PRIO: 3
Time slot   7
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  1
        Loaded a process at input/proc/m0s, PID: 5 PRIO: 3
Time slot   8
Time slot   9
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  3
        Loaded a process at input/proc/p1s, PID: 6 PRIO: 2
Time slot  10
Time slot  11
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  4
        Loaded a process at input/proc/s0, PID: 7 PRIO: 1
Time slot  12
Time slot  13
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot  14
```

```
Time slot  15
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot  16
        Loaded a process at input/proc/s1, PID: 8 PRIO: 0
Time slot  17
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot  18
Time slot  19
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot  20
Time slot  21
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  4
Time slot  22
Time slot  23
        CPU 0: Processed  4 has finished
        CPU 0: Dispatched process  2
Time slot  24
Time slot  25
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  2
Time slot  26
Time slot  27
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  2
Time slot  28
Time slot  29
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  2
Time slot  30
Time slot  31
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  2
Time slot  32
        CPU 0: Processed  2 has finished
        CPU 0: Dispatched process  5
Time slot  33
Time slot  34
        CPU 0: Put process  5 to run queue
        CPU 0: Dispatched process  1
Time slot  35
Time slot  36
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  8
Time slot  37
Time slot  38
        CPU 0: Put process  8 to run queue
```

```
        CPU 0: Dispatched process  7
Time slot  39
Time slot  40
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  3
Time slot  41
Time slot  42
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  3
Time slot  43
Time slot  44
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  3
Time slot  45
Time slot  46
        CPU 0: Processed  3 has finished
        CPU 0: Dispatched process  6
Time slot  47
Time slot  48
        CPU 0: Put process  6 to run queue
        CPU 0: Dispatched process  5
Time slot  49
Time slot  50
        CPU 0: Put process  5 to run queue
        CPU 0: Dispatched process  1
Time slot  51
        CPU 0: Processed  1 has finished
        CPU 0: Dispatched process  8
Time slot  52
Time slot  53
        CPU 0: Put process  8 to run queue
        CPU 0: Dispatched process  7
Time slot  54
Time slot  55
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  6
Time slot  56
Time slot  57
        CPU 0: Put process  6 to run queue
        CPU 0: Dispatched process  5
write region=1 offset=20 value=102
print_pgtbl: 0 - 512
00000000: 80000003
00000004: 80000002
Time slot  58
write region=2 offset=1000 value=1
print_pgtbl: 0 - 512
00000000: 80000003
00000004: 80000002
Time slot  59
```

```
        CPU 0: Put process  5 to run queue
        CPU 0: Dispatched process  8
Time slot  60
Time slot  61
        CPU 0: Put process  8 to run queue
        CPU 0: Dispatched process  7
Time slot  62
Time slot  63
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  6
Time slot  64
Time slot  65
        CPU 0: Put process  6 to run queue
        CPU 0: Dispatched process  5
write region=0 offset=0 value=0
print_pgtbl: 0 - 512
00000000: 80000003
00000004: c0000000
Time slot  66
        CPU 0: Processed  5 has finished
        CPU 0: Dispatched process  8
Time slot  67
        CPU 0: Processed  8 has finished
        CPU 0: Dispatched process  7
Time slot  68
Time slot  69
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  6
Time slot  70
Time slot  71
        CPU 0: Put process  6 to run queue
        CPU 0: Dispatched process  7
Time slot  72
Time slot  73
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  6
Time slot  74
Time slot  75
        CPU 0: Processed  6 has finished
        CPU 0: Dispatched process  7
Time slot  76
Time slot  77
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  7
Time slot  78
Time slot  79
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  7
Time slot  80
        CPU 0: Processed  7 has finished
```

```
185         CPU 0 stopped
```

2.2.3 Giải thích trạng thái RAM

- **Time slot 6** :CPU executes instruction alloc 300 0 of m1s. Result:

```
1         ===== PHYSICAL MEMORY AFTER ALLOCATION =====
2      PID=3 - Region=0 - Address=00000000 - Size=300 byte
3      print_pgtbl: 0 - 512
4      00000000: 80000001
5      00000004: 80000000
6      Page Number: 0 -> Frame Number: 1
7      Page Number: 1 -> Frame Number: 0
8      ================================================================
```

- **Time slot 7:** CPU executes instruction alloc 100 1 of m1s. Result:

```
1       ===== PHYSICAL MEMORY AFTER ALLOCATION =====
2   PID=3 - Region=1 - Address=0000012c - Size=100 byte
3   print_pgtbl: 0 - 512
4   00000000: 80000001
5   00000004: 80000000
6   Page Number: 0 -> Frame Number: 1
7   Page Number: 1 -> Frame Number: 0
8   ================================================================
```

- **Time slot 8:** CPU executes instruction free 0 of m1s. Result:

```
1       ===== PHYSICAL MEMORY AFTER ALLOCATION =====
2   PID=3 - Region=0
3   print_pgtbl: 0 - 512
4   00000000: 80000001
5   00000004: 80000000
6   Page Number: 0 -> Frame Number: 1
7   Page Number: 1 -> Frame Number: 0
8   ================================================================
```

- **Time slot 9:** CPU executes instruction alloc 100 2 of m1s. Result:

```
1   ===== PHYSICAL MEMORY AFTER ALLOCATION =====
2   PID=3 - Region=2 - Address=00000000 - Size=100 byte
3   print_pgtbl: 0 - 512
4   00000000: 80000001
5   00000004: 80000000
6   Page Number: 0 -> Frame Number: 1
7   Page Number: 1 -> Frame Number: 0
8   ================================================================
```

- **Time slot 34:** CPU executes instruction free 2 of m1s. Result:

```
===== PHYSICAL MEMORY AFTER DEALLOCATION =====
PID=3 - Region=2
print_pgtbl: 0 - 512
00000000: 80000001
00000004: 80000000
Page Number: 0 -> Frame Number: 1
Page Number: 1 -> Frame Number: 0
==================================
================================================================
```

- **Time slot 35:** CPU executes instruction free 1 of m1s. Result:

```
    ===== PHYSICAL MEMORY AFTER DEALLOCATION =====
PID=3 - Region=1
print_pgtbl: 0 - 512
00000000: 80000001
00000004: 80000000
Page Number: 0 -> Frame Number: 1
Page Number: 1 -> Frame Number: 0
================================================================
```

- **Time slot 48:** CPU executes instruction alloc 300 0 of m0s. Result:

```
    ===== PHYSICAL MEMORY AFTER ALLOCATION =====
PID=5 - Region=0 - Address=00000000 - Size=300 byte
print_pgtbl: 0 - 512
00000000: 80000003
00000004: 80000002
Page Number: 0 -> Frame Number: 3
Page Number: 1 -> Frame Number: 2
================================================================
```

- **Time slot 49:** CPU executes instruction alloc 100 1 of m0s. Result:

```
    ===== PHYSICAL MEMORY AFTER ALLOCATION =====
PID=5 - Region=1 - Address=0000012c - Size=100 byte
print_pgtbl: 0 - 512
00000000: 80000003
00000004: 80000002
Page Number: 0 -> Frame Number: 3
Page Number: 1 -> Frame Number: 2
================================================================
```

- **Time slot 54:** CPU executes instruction free 0 of m0S. Result:

```
    ===== PHYSICAL MEMORY AFTER DEALLOCATION =====
PID=5 - Region=0
print_pgtbl: 0 - 512
00000000: 80000003
00000004: 80000002
Page Number: 0 -> Frame Number: 3
Page Number: 1 -> Frame Number: 2
================================================================
```

- **Time slot 55:** CPU executes instruction alloc 100 2 of m0s. Result:

```
    ===== PHYSICAL MEMORY AFTER ALLOCATION =====
PID=5 - Region=2 - Address=00000000 - Size=100 byte
print_pgtbl: 0 - 512
00000000: 80000003
00000004: 80000002
Page Number: 0 -> Frame Number: 3
Page Number: 1 -> Frame Number: 2
================================================================
```

- **Time slot 60:** CPU executes instruction write 102 1 20 of m0s. Result:

```
    ===== PHYSICAL MEMORY AFTER WRITING =====
write region=1 offset=20 value=102
print_pgtbl: 0 - 512
00000000: 80000003
00000004: 80000002
Page Number: 0 -> Frame Number: 3
Page Number: 1 -> Frame Number: 2
================================================================
===== PHYSICAL MEMORY DUMP =====
BYTE 00000240: 102
===== PHYSICAL MEMORY END-DUMP =====
================================================================
```

- **Time slot 61:** CPU executes instruction write 1 2 1000 of m0s. Result:

```
    ===== PHYSICAL MEMORY AFTER WRITING =====
write region=2 offset=1000 value=1
print_pgtbl: 0 - 512
00000000: c0000000
00000004: 80000002
Page Number: 0 -> Frame Number: 0
Page Number: 1 -> Frame Number: 2
================================================================
===== PHYSICAL MEMORY DUMP =====
BYTE 00000240: 102
BYTE 000003e8: 1
===== PHYSICAL MEMORY END-DUMP =====
```

```
13 ================================================================
```