TRƯỜNG ĐẠI HỌC BÁCH KHOA TP HCM
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

HỆ ĐIỀU HÀNH

Báo cáo Bài tập lớn số 02

Simple Operating System

GVHD: Nguyễn Hoài Nam
SV thực hiện: 1712451 Trần Minh Nhân
1810998 Nguyễn Huỳnh Hữu Khiêm
1810995 Hồ Quang Khải
1811389 Sỳ Tùng An

Tp. Hồ Chí Minh, Tháng 06/2020

Mục lục

1 Trả lời câu hỏi

1.1 Question 1: Ưu điểm của việc sử dụng hàng đợi ưu tiên so với các giải thuật sắp xếp định thời đã học là gì?

Độ ưu tiên của process khi được xác định bên trong, có thể được quyết định dựa trên yêu cầu bộ nhớ, giới hạn thời gian, số lượng tệp đang mở, tỷ lệ của burst I / O với burst CPU, v.v.
Trong khi đó, các ưu tiên bên ngoài được đặt dựa trên các tiêu chí bên ngoài hệ điều hành, như tầm quan trọng của quy trình, tiền được trả cho việc sử dụng tài nguyên máy tính, yếu tố makrte, v.v.
Điều này cung cấp một cơ chế tốt trong đó tầm quan trọng tương đối của mỗi quá trình có thể được xác định chính xác.

Giải thuật Priority Feedback Queue (PFQ) sử dụng tư tưởng của một số giải thuật khác gồm giaỉ thuật Priority Scheduling - mỗi process mang một độ ưu tiên để thực thi, giải thuật Multilevel Queue - sử dụng nhiều mức hàng đợi các process, giải thuật Round Robin - sử dụng quantum time cho các process thực thi. Dưới đây là các giải thuật định thời khác đã học:

- First Come First Served (FCFS)
- Shortest Job First (SJF)
- Shortest Remaining Time First (SRTF)
- Priority Scheduling (PS)
- Round Robin (RR)
- Multilevel Queue Scheduling (MLQS)
- Multilevel Feedback Queue (MLFQ)

Cụ thể, giải thuật PFQ sử dụng 2 hàng đợi là $ready\_queue$ và run_queue có ý nghĩa:

- $ready\_queue$: hàng đợi chứa các process ở mức độ ưu tiên thực thi cao hơn so với hàng đợi run_queue. Khi CPU chuyển sang lượt tiếp theo, nó sẽ tìm kiếm process ở hàng đợi này.
- run_queue: hàng đợi chứa các process đang chờ để tiếp tục thực thi sau khi hết lượt của nó mà chưa hoàn thành quá trình. Các process ở hàng đợi này chỉ được tiếp tục thực hiện tiếp theo khi $ready\_queue$ lỗi và được đưa sang hàng đợi $ready\_queue$ để xét lượt tiếp theo.
- Cả hai hàng đợi đều là hàng đợi có độ ưu tiên, mức độ ưu tiên dựa trên mức độ ưu tiên của process trong hàng đợi.

Ưu điểm của giải thuật PFQ

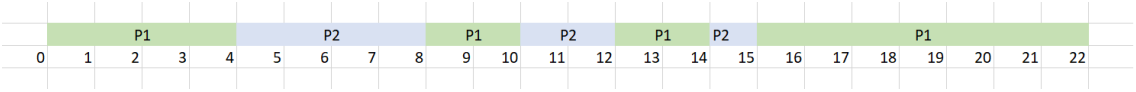
- Sử dụng time slot, mang tư tưởng của giải thuật RR với một khoảng quantum time, tạo sự công bằng về thời gian thực thi giữa các process, tránh tình trạng chiếm CPU sử dụng, trì hoãn vô hạn định.
- Sử dụng hai hàng đợi, mang tư tưởng của giải thuật MLQS và MLFQ, trong đó hai hàng đợi được chuyển qua lại các process đến khi process được hoàn tất, tăng thời gian đáp ứng cho các process (các process có độ ưu tiên thấp đến sau vẫn có thể được thực thi trước các process có độ ưu tiên cao hơn sau khi đã xong phần của mình).
- Tính công bằng giữa các process là được đảm bảo, chỉ phụ thuộc vào độ ưu tiên có sẵn của các process. Cụ thể xét trong khoảng thời gian $t0$ nào đó, nếu các process đang thực thi thì hoàn toàn phụ thuộc vào độ ưu tiên của chúng. Nếu có 1 process $p0$ khác đến, giả sử $ready\_queue$ đang sẵn sàng, process $p0$ này vào hàng đợi ưu tiên và phụ thuộc vào độ ưu tiên của nó, cho dù trước đó các process khác có độ ưu tiên cao hơn đã thực thi xong, chúng cũng không thể tranh chấp với process $p0$ được vì chúng đang chờ trong run_queue cho đến khi $ready\_queue$ lỗi, tức $p0$ đã được thực thi lượt của nó.

1.2 Question 2: Hãy cho biết thuận lợi và bất lợi của việc phân đoạn với phân trang.

- **Thuận lợi**
  - Đơn giản để di chuyển các phân đoạn hơn là toàn bộ không gian địa chỉ.
  - Không có sự phân mảnh bên trong như sự phân mảnh bên ngoài.
  - Bảng phân đoạn có kích thước nhỏ hơn so với bảng trang trong phân trang.
  - Kích thước trung bình của phân đoạn lớn hơn kích thước thực của trang
  - Đưa ra sự bảo vệ trong phân đoạn
  - Bảng phân đoạn sử dụng bộ nhớ ít hơn so với phân trang
  - Vì nó cung cấp bảng phân đoạn nhỏ, tham chiếu bộ nhớ rất đơn giản, nên có thể cho mượn để chia sẻ dữ liệu giữa các quy trình.
- **Bất lợi**
  - Kích thước không bằng nhau của các phân đoạn khiến nó trở nên không tốt trong trường hợp hoán đổi.
  - Việc chuyển từ Linux sang các kiến trúc khác nhau rất khó xử lý vì nó cung cấp hỗ trợ rất hạn chế cho phân đoạn.
  - Đòi hỏi sự can thiệp của lập trình viên.
  - Thật khó để phân bổ bộ nhớ lan truyền cho phân vùng vì nó có kích thước thay đổi.
  - Đây là thuật toán quản lý bộ nhớ tốn kém.

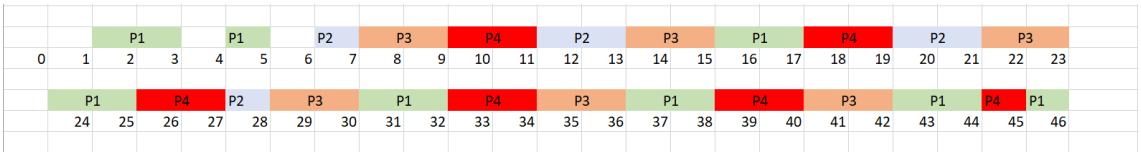
2 Scheduling

2.1 Sched0



Hình 1: *Lược đồ Gantt CPU thực thi các processes - test 0*

2.2 Sched1



Hình 2: *Lược đồ Gantt CPU thực thi các processes - test 1*

3 Memory

Dưới đây là kết quả của quá trình ghi log sau mỗi lệnh allocation và deallocation trong chương trình, cụ thể ghi lại trạng thái của RAM trong chương trình ở mỗi bước.

3.1 Test 0

:

```
============ Allocation ============
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
============ Allocation ============
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
============ Deallocation ============
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
============ Allocation ============
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
============ Allocation ============
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
```

```
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)

============ Final - dump() ============
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
        003e8: 15
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
        03814: 66
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
```

3.2 Test 1

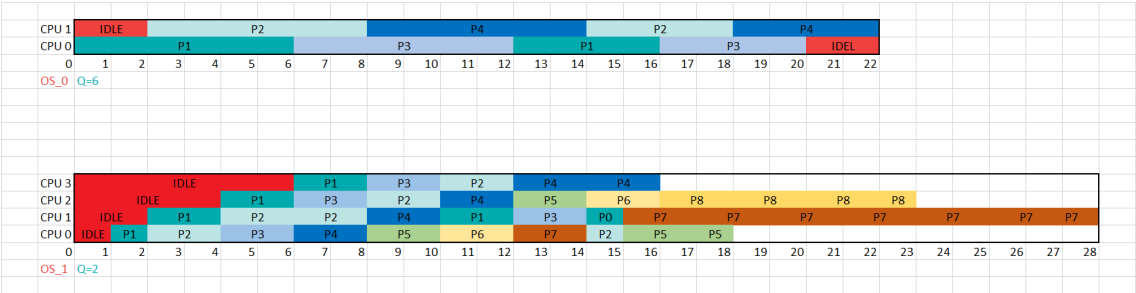
:

```
========== Allocation ==========
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
========== Allocation ==========
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
========== Deallocation ==========
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
========== Allocation ==========
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
========== Allocation ==========
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
```

```
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
============  Deallocation  ============
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
============  Deallocation  ============
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
============  Deallocation  ============

============  Final - dump()  ============
```

4 Overall



Hình 3: *Lược đồ Gantt CPU thực thi các processes cho make all*