**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN ĐIỆN TỬ VIỄN THÔNG**

A picture containing icon

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN HỆ ĐIỀU HÀNH**

**ĐỊNH THỜI CPU (SCHEDULING)**

**Giảng viên hướng dẫn**: TS. Hàn Huy Dũng

**Nhóm thực hiện**: Nhóm 11

**Mã lớp:** 129266

|  |  |
| --- | --- |
| **Họ và tên** | **MSSV** |
| Dương Công Kiên | 20182614 |
| Vũ Mạnh Cường | 20182404 |
| Lê Hoàng Anh | 20182337 |

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 3](#_Toc96501625)

[**CHƯƠNG I: CÁC KHÁI NIỆM CƠ BẢN** 4](#_Toc96501626)

[**1.** **Đặt vấn đề** 4](#_Toc96501627)

[**2.** **Các khái niệm** 4](#_Toc96501628)

[**2.1.** **Trình lập lịch CPU – CPU Scheduler** 5](#_Toc96501629)

[**2.2.** **Preemptive/nonpreemptive Scheduling** 5](#_Toc96501630)

[**2.3.** **Trình điều vận – Dispatcher** 5](#_Toc96501631)

[**3.** **Phân loại các hoạt động định thời** 6](#_Toc96501632)

[**4.** **Hai thành phần của chiến lược định thời** 6](#_Toc96501633)

[**4.1.** **Hàm lựa chọn** 6](#_Toc96501634)

[**4.2.** **Chế độ quyết định (Decision Mode)** 6](#_Toc96501635)

[**4.3.** **Thời điểm thực thi *Selection Function (SF)*** 6](#_Toc96501636)

[**CHƯƠNG II: CÁC TIÊU CHUẨN LẬP LỊCH** 7](#_Toc96501637)

[**CHƯƠNG III: CÁC GIẢI THUẬT LẬP LỊCH** 8](#_Toc96501638)

[**1.** **Giải thuật First-Come, First-Served (FCFS)** 8](#_Toc96501639)

[**2.** **Giải thuật Shortest-Job-First (SJF)** 9](#_Toc96501640)

[**2.1.** **Xác định thời gian sử dụng CPU tiếp sau** 10](#_Toc96501641)

[**3.** **Lập lịch theo mức ưu tiên** 11](#_Toc96501642)

[**4.** **Giải thuật Round-Robin (RR)** 12](#_Toc96501643)

[**4.1.** **Quan hệ giữa q và hiệu năng** 13](#_Toc96501644)

[**4.2.** **Phụ thuộc của Turnaround Time vào q** 13](#_Toc96501645)

[**5.** **Lập lịch đa mức hàng đợi** 14](#_Toc96501646)

[**5.1.** **Multilevel Queue Scheduling** 14](#_Toc96501647)

[**6.** **Lập lịch đa mức hàng đợi có hoàn ngược** 15](#_Toc96501648)

[**CHƯƠNG IV: CÁC LOẠI LẬP LỊCH** 17](#_Toc96501649)

[**1.** **Lập lịch multiprocessor** 17](#_Toc96501650)

[**2.** **Lập lịch thời gian thực** 17](#_Toc96501651)

[**3.** **Lập lịch luồng** 17](#_Toc96501652)

[**CHƯƠNG V: VÍ DỤ LẬP LỊCH TRONG HĐH LINUX** 18](#_Toc96501653)

[**1.** **Linux scheduling** 18](#_Toc96501654)

[**2.** **Quan hệ giữa Priorities và Time-slice length** 18](#_Toc96501655)

[**3.** **Linux Scheduling (tiếp)** 18](#_Toc96501656)

[**CHƯƠNG VI: LỰA CHỌN GIẢI THUẬT** 20](#_Toc96501657)

[**CHƯƠNG VII: Mô phỏng và đánh giá một số giải thuật Scheduling** 21](#_Toc96501658)

[**1.** **Một số thông số cần lưu ý** 21](#_Toc96501659)

[**2.** **Mô phỏng một số giải thuật** 21](#_Toc96501660)

[**2.1. First Come – First Served (FCFS)** 22](#_Toc96501661)

[**2.2. Round – Robin (RR)** 22](#_Toc96501662)

[**2.3. Short Job First (SJF)** 23](#_Toc96501663)

[**CHƯƠNG VIII: ĐÁNH GIÁ TRÊN LINUX** 25](#_Toc96501664)

[**1.** **Bài toán đặt ra:** 25](#_Toc96501665)

[**2.** **Trường hợp 1: SCHED\_OTHER; NI thay đổi** 25](#_Toc96501666)

[**3.** **Trường hợp 2: Thay đổi Scheduling Policies** 26](#_Toc96501667)

[**4.** **Trường hợp 3: Tương tự TH2, thay đổi NI** 27](#_Toc96501668)

[**5.** **Theo dõi sự khác biệt trên Web Server:** 29](#_Toc96501669)

[**6.** **Nhận xét các kết quả tổng quan:** 30](#_Toc96501670)

[**CHƯƠNG IX: TỔNG KẾT** 31](#_Toc96501671)

[**TÀI LIỆU THAM KHẢO** 32](#_Toc96501672)

**MỤC LỤC HÌNH ẢNH**

[Hình 1: Sơ đồ trạng thái 6](#_Toc96501680)

[Hình 2: Các hoạt động định thời 7](#_Toc96501681)

[Hình 3: Sơ đồ trạng thái 8](#_Toc96501682)

[Hình 4: SJF không ưu tiên trước 11](#_Toc96501683)

[Hình 5: SJF có ưu tiên trước 11](#_Toc96501684)

[Hình 6: Minh họa ví dụ 12](#_Toc96501685)

[Hình 7: Ví dụ lập lịch theo mức ưu tiên 13](#_Toc96501686)

[Hình 8: Ví dụ lập lịch RR 14](#_Toc96501687)

[Hình 9: Quan hệ giữa q và hiệu năng 14](#_Toc96501688)

[Hình 10:Phụ thuộc của Turnaround Time vào q 15](#_Toc96501689)

[Hình 11: Multilevel Queue Scheduling 16](#_Toc96501690)

[Hình 12: Multilevel Feedback Queues 17](#_Toc96501691)

[Hình 13: Quan hệ giữa Priorities và Time-slice length 19](#_Toc96501692)

[Hình 14: Mô phỏng giải thuật 22](#_Toc96501693)

[Hình 15: First come - first serve 23](#_Toc96501694)

[Hình 16: RR Qt = 2ms 24](#_Toc96501695)

[Hình 17: RR Qt = 20ms 24](#_Toc96501696)

[Hình 18: SJF 25](#_Toc96501697)

[Hình 19: thời gian sử dụng tài nguyên 27](#_Toc96501698)

[Hình 20: Thời gian sử dụng tài nguyên trong 5 phút của tiến trình 27](#_Toc96501699)

[Hình 21: Thời gian sử dụng tài nguyên 28](#_Toc96501700)

[Hình 22:Thời gian sử dụng CPU theo SCHED 28](#_Toc96501701)

[Hình 23: thời gian sử dụng tài nguyên 29](#_Toc96501702)

[Hình 24: thời gian sử dụng CPU theo NI, SCHED 29](#_Toc96501703)

[Hình 25: Theo dõi sự khác biệt trên Webserver 30](#_Toc96501704)

[Hình 26: Theo dõi thời gian phản hồi của Webserver 31](#_Toc96501705)

# **LỜI NÓI ĐẦU**

Hệ điều hành (Operating System) là một thành phần không thể thiếu trong một hệ thống máy tính. Một máy tính dù đắt tiền, cấu hình cao nếu như không có hệ điều hành thì hầu như không thể sử dụng được. Hệ điều hành điều khiển mọi hoạt động của máy tính, giúp việc sử dụng máy tính trở nên đơn giản, dễ dàng và hiệu quả hơn nhiều. Do vậy việc môn học “Hệ điều hành” là một môn học vô cùng quan trọng và cần thiết trong chương trình đào tạo chuyên ngành điện tử viễn thông, ngành cần giao tiếp nhiều với phần cứng (bao gồm cả máy tính)

Trong suốt học kỳ vừa qua, với sự giúp đỡ của TS. Hàn Huy Dũng, nhóm 11 chúng em đã trải qua nhiều lần thuyết trình với các tìm hiểu khác nhau về hệ điều hành Linux. Sau 1 khoảng thời gian tích lũy kiến thức cũng như tìm hiểu thêm ở các nguồn khác, chúng em đã quyết định lựa chọn đề tài bài tập lớn cuối kỳ kết thúc môn học là: “Định thời CPU – CPU Scheduling”

Tuy rằng có thể báo cáo còn đôi chút thiếu sót, chúng em mong thầy và các bạn có những góp ý để bài tập của nhóm 11 chúng em được hoàn thiện hơn nữa!

# **CHƯƠNG I: CÁC KHÁI NIỆM CƠ BẢN**

1. **Đặt vấn đề**

Trong các hệ thống Multiprogramming / Multitasking:

* Tại một thời điểm, trong bộ nhớ có nhiều Process
* Tại mỗi thời điểm chỉ có một Process được thực thi (Giả sử hệ thống chỉ có 1 CPU (1 Processor))
* Do đó, cần phải giải quyết vấn đề phân loại và lựa chọn Process thực thi sao cho được hiệu quả nhất (theo các tiêu chí định thời). Cần có chiến lược định thời CPU (Scheduling)

1. **Các khái niệm**
   1. **Trình lập lịch CPU – CPU Scheduler**

* Mỗi khi CPU rỗi, HĐH cần chọn trong số các tiến trình đã sẵn sàng thực hiện trong bộ nhớ (ready queue), và phân phối CPU cho một trong số đó.
* Tiến trình được thực hiện bởi trình lập lịch ngắn kỳ (short-term scheduler, CPU scheduler)
* Các quyết định lập lịch CPU có thể xảy ra khi một tiến trình:
* Chuyển từ trạng thái chạy sang trạng thái chờ (vd: I/O request)
* Chuyển từ trạng thái chạy sang trạng thái sẵn sàng (vd: khi một ngắt xuất hiện)
* Chuyển từ trạng thái đợi sang trạng thái sẵn sàng (vd: I/O hoàn thành)
* Kết thúc

Diagram

Description automatically generated

Hình 1: Sơ đồ trạng thái

* 1. **Preemptive/nonpreemptive Scheduling**
* Lập lịch CPU khi 1 và 4 là *không được ưu tiên trước* (*nonpreemptive*):
* Không có sự lựa chọn: phải chọn 1 tiến trình mới để thực hiện.
* Khi 1 tiến trình được phân phối CPU, nó sẽ sử dụng CPU cho đến khi nó giải phóng CPU bằng cách kết thúc hoặc chuyển sang trạng thái chờ.
* Các tiến trình sẵn sàng nhường điều khiển của CPU.
* Lập lịch khi 2 và 3 là *được ưu tiên trước* (*preemptive*)
* Khi 2: tiến trình đá bật CPU ra. Cần phải chọn tiến trình kế tiếp.
* Khi 3: tiến trình có thể đá bật tiến trình khác ra khỏi CPU.
  1. **Trình điều vận – Dispatcher**
* Môđun trình điều vận trao quyền điều khiển của CPU cho tiến trình được lựa chọn bởi trình lập lịch CPU, các bước:
* chuyển ngữ cảnh
* chuyển sang user mode
* nhảy tới vị trí thích hợp trong chương trình của người sử dụng để khởi động lại chương trình đó
* *Trễ điều vận (Dispatch latency)* – thời gian cần thiết để trình điều vận dừng một tiến trình và khởi động một tiến trình khác chạy.

1. **Phân loại các hoạt động định thời**

Diagram

Description automatically generated

Hình 2: Các hoạt động định thời

1. **Hai thành phần của chiến lược định thời**
   1. **Hàm lựa chọn**

* Xác định Process vào Ready queue sẽ được thực thi tiếp theo. Thường theo các tiêu chí sau:
* w = tổng thời gian thực thi của hệ thống
* e = thời gian đã được phục vụ
* s = tổng thời gian thực thi của Process (bao gồm cả giá trị e)
  1. **Chế độ quyết định (Decision Mode)**
* Định nghĩa thời điểm hàm lựa chọn được thực thi:
* *Nonpreemptive*: Một Process ở trạng thái running cho đến khi nó block hoặc terminated
* *Preemtive*: Process đang thực thi có thể bị ngắt và chuyển về trạng thái Ready. Tránh trường hợp một Process độc chiếm CPU.
  1. **Thời điểm thực thi *Selection Function (SF)***

Diagram, schematic

Description automatically generated

Hình 3: Sơ đồ trạng thái

* *Nonpreemptive:* chỉ thực thi SF trong trường hợp (1) và (4): quá trình running nếu bị ngắt sẽ tiếp tục running sau đó.
* *Preemptive: ngoài trường hợp (1) và (4) còn thực thi thêm SF trong trường hợp (2) hoặc (3) (hoặc đồng thời cả (2) và (3) ).*

# **CHƯƠNG II: CÁC TIÊU CHUẨN LẬP LỊCH**

**CPU utilization** – giữ cho CPU càng bận càng tốt (0-100%)

**Throughput** – số tiến trình được hoàn thành trong một đơn vị thời gian

**Turnaround time** – tổng lượng thời gian để thực hiện một tiến trình: t/g chờ được đưa vào bộ nhớ + t/g chờ trong ready queue + t/g thực hiện bởi CPU + t/g thực hiện vào-ra

**Waiting time** – lượng thời gian mà một tiến trình chờ đợi ở trong ready queue

**Response time** – lượng thời gian tính từ khi có một yêu cầu được gửi đến khi có sự trả lời đầu tiên được phát ra, không phải là thời gian đưa ra kết quả của sự trả lời đó. → là tiêu chuẩn tốt.

# **CHƯƠNG III: CÁC GIẢI THUẬT LẬP LỊCH**

1. **Giải thuật First-Come, First-Served (FCFS)**

* Tiến trình nào yêu cầu CPU trước sẽ được phân phối CPU trước   
  → Giải thuật FCFS là không được ưu tiên trước.
* Là giải thuật đơn giản nhất

|  |  |
| --- | --- |
| Process | Burst Time (thời gian sử dụng CPU, ms) |
| P1 | 24 |
| P2 | 3 |
| P3 | 3 |

* Giả định rằng các tiến trình đến theo thứ tự: *P1*, *P2*, *P3* thìbiểu đồ Gantt (Gantt Chart) của lịch biểu như sau:

0

24

27

30

|  |  |  |
| --- | --- | --- |
| P1 | P2 | P3 |

* Thời gian chờ đợi của các tiến trình: *P1* = 0; *P2* = 24; *P3* = 27
* Thời gian chờ đợi trung bình: (0 + 24 + 27)/3 = 17
* Giả định rằng các tiến trình đến theo thứ tự *P2* , *P3* , *P1*
* Biểu đồ Gantt của lịch biểu như sau:

30

6

3

0

|  |  |  |
| --- | --- | --- |
| P2 | P3 | P1 |

* Thời gian chờ đợi của các tiến trình: *P1 =* 6;*P2* = 0;*; P3 =* 3
* Thời gian chờ đợi trung bình: (6 + 0 + 3)/3 = 3
* Tốt hơn nhiều so với trường hợp trước
* *Convoy effect* (hiệu ứng hộ tống): tiến trình ngắn đứng sau tiến trình dài, như là các xe máy đi sau xe buýt vậy.

1. **Giải thuật Shortest-Job-First (SJF)**

* Gắn với mỗi tiến trình là thời gian sử dụng CPU *tiếp sau* của nó. Thời gian này được sử dụng để lập lịch các tiến trình với thời gian ngắn nhất.
* Hai phương pháp:
* không ưu tiên trước (non-preemptive)– một tiến trình nếu sử dụng CPU thì không nhường cho tiến trình khác cho đến khi nó kết thúc
* có ưu tiên trước (preemptive)– nếu một tiến trình đến có thời gian sử dụng CPU *ngắn hơn* thời gian còn lại của tiến trình đang thực hiện thì ưu tiên tiến trình mới đến trước. Phương pháp này còn được gọi là Shortest-Remaining-Time-First (SRTF).
* SJF là tối ưu – cho thời gian chờ đợi trung bình của các tiến trình là nhỏ nhất.

VD: **Ví dụ SJF không ưu tiên trước**

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 7 |
| P2 | 2 | 4 |
| P3 | 4 | 1 |
| P4 | 5 | 4 |

* SJF (non-preemptive)

Timeline

Description automatically generated

Hình 4: SJF không ưu tiên trước

* Thời gian chờ đợi của các tiến trình: *P1 =* 0;*P2* = 6;*; P3 =* 3, *P4 =* 7
* Thời gian chờ đợi trung bình = (0 + 6 + 3 + 7)/4 = 4

VD: **Ví dụ SJF có ưu tiên trước**

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 7 |
| P2 | 2 | 4 |
| P3 | 4 | 1 |
| P4 | 5 | 4 |

* SJF (non-preemptive)

A picture containing text, clock

Description automatically generated

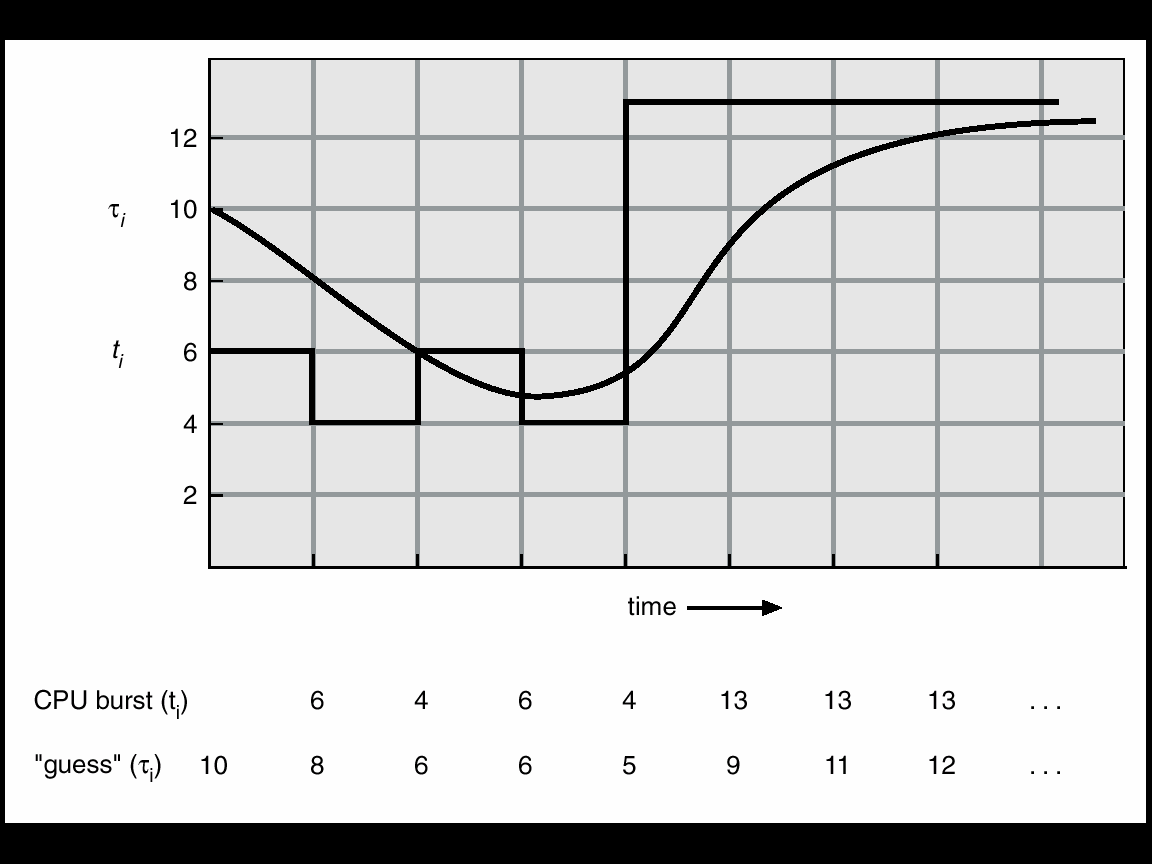
Hình 5: SJF có ưu tiên trước

* Thời gian chờ đợi trung bình = (9 + 1 + 0 +2)/4 = 3
  1. **Xác định thời gian sử dụng CPU tiếp sau**
* Không thể biết chính xác thời gian sử dụng CPU tiếp sau của tiến trình nhưng có thể đoán giá trị xấp xỉ của nó dựa vào thời gian sử dụng CPU trước đó và sử dụng công thức đệ quy



* τn+1 = giá trị dự đoán cho thời gian sử dụng CPU tiếp sau
* tn = thời gian thực tế của sự sử dụng CPU thứ n
* α, 0 **≤** α **≤** 1
* τ0 là một hằng số
* α = 0: τn+1 = τn = τ0.
* Thời gian thực tế sử dụng CPU gần đây không có tác dụng gì cả.
* α = 1: τn+1 = *t*n.
* Chỉ tính đến thời gian sử dụng CPU thực tế ngay trước đó.

VD: **Minh họa khi α = 1/2 và τ0 = 10**

****

Hình 6: Minh họa ví dụ

1. **Lập lịch theo mức ưu tiên**

* Mỗi tiến trình được gắn một số ưu tiên (số nguyên). VD: 0-127
* CPU được phân phối cho tiến trình có mức ưu tiên cao nhất (có số ưu tiên nhỏ nhất)
* Preemptive
* Nonpreemptive
* SJF là trường hợp riêng của lập lịch theo mức ưu tiên: mức ưu tiên chính là thời gian sử dụng CPU tiếp sau dự đoán được.
* Vấn đề gặp phải là: những tiến trình có mức ưu tiên thấp có thể không bao giờ được thực hiện (starvation).
* Giải pháp ≡ Aging: kỹ thuật tăng mức ưu tiên của các tiến trình chờ đợi lâu trong hệ thống.

VD: Sau 1-15 phút giảm số ưu tiên một lần

* VD: **Ví dụ lập lịch theo mức ưu tiên**

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 4 |
| P4 | 1 | 5 |
| P5 | 5 | 2 |

* Nonpreemptive:

Text

Description automatically generated

Hình 7: Ví dụ lập lịch theo mức ưu tiên

* Thời gian chờ đợi trung bình = (0 + 1 + 6 + 16 + 18)/5 = 8.2

1. **Giải thuật Round-Robin (RR)**

* Mỗi tiến trình sử dụng một lượng nhỏ thời gian của CPU (*time quantum – định lượng thời gian, q*), thường là 10-100 ms. Sau đó nó được ưu tiên đưa vào cuối của ready queue.
* Ready queue được tổ chức dạng FIFO (FCFS)
* Nếu tiến trình có thời gian sử dụng CPU < *q* ⇒ Tiến trình sẽ tự nguyện nhường CPU khi kết thúc. Trình lập lịch sẽ chọn tiến trình kế tiếp trong ready queue.
* Nếu tiến trình có thời gian sử dụng CPU > *q* ⇒ bộ định thời (timer) sẽ đếm lùi và gây ngắt HĐH khi nó = 0. Việc chuyển ngữ cảnh được thực hiện và tiến trình hiện tại được đưa xuống cuối ready queue để nhường CPU cho tiến trình kế tiếp.
* VD: **Ví dụ lập lịch RR với q = 20**

|  |  |
| --- | --- |
| Process | Burst Time |
| P1 | 53 |
| P2 | 17 |
| P3 | 68 |
| P4 | 24 |

* Biểu đồ Gantt:

A picture containing text, clock

Description automatically generated

Hình 8: Ví dụ lập lịch RR

* Thời gian chờ đợi TB = ((57+24)+20+(37+40+17)+(57+40))/4 = 73
* Thường thì *RR* cóturnaround trung bình cao hơn SJF, nhưng có *response* tốt hơn (thấp hơn).
  1. **Quan hệ giữa q và hiệu năng**
* Nếu q lớn ⇒ tương tự như FCFS.
* Nếu q nhỏ ⇒ số lần chuyển ngữ cảnh càng nhiều, làm giảm hiệu năng.

Diagram

Description automatically generated with medium confidence

Hình 9: Quan hệ giữa q và hiệu năng

* 1. **Phụ thuộc của Turnaround Time vào q**
* Turnaround Time trung bình của một tập tiến trình không phải luôn được cải thiện khi q tăng lên.
* Luật: 80% các CPU burst nên nhỏ hơn q.

Chart, line chart

Description automatically generated

Hình 10:Phụ thuộc của Turnaround Time vào q

1. **Lập lịch đa mức hàng đợi**

* Ready queue được chia thành nhiều queue riêng biệt:
* foreground (chứa các *interactive process*)
* background (chứa các *batch process*)
* Mỗi hàng đợi có giải thuật lập lịch riêng:
* foreground – RR
* background – FCFS
* Phải có sự lập lịch giữa các queue:
* Lập lịch với mức ưu tiên cố định; vd: phục vụ tất cả tiến trình từ foreground, tiếp theo từ background (có thể xảy ra starvation).
* Phân chia thời gian: mỗi queue nhận được một lượng thời gian CPU nào đó mà nó có thể lập lịch các tiến trình của nó.

VD: 80% cho foreground và 20% cho background queue

* 1. **Multilevel Queue Scheduling**

Chart

Description automatically generated

Hình 11: Multilevel Queue Scheduling

* Tiến trình trong queue có mức ưu tiên thấp hơn chỉ có thể chạy khi các queue có mức ưu tiên cao hơn rỗng.
* Tiến trình có mức ưu tiên cao hơn khi vào ready queue không ảnh hưởng đến tiến trình đang chạy có mức ưu tiên thấp hơn.

1. **Lập lịch đa mức hàng đợi có hoàn ngược**

* Một tiến trình có thể di chuyển giữa các queue khác nhau; có thể thực hiện aging
* Trình lập lịch đa mức hàng đợi có hoàn ngược được xác định bởi các tham số sau:
* số lượng queue
* giải thuật lập lịch cho mỗi queue
* phương pháp được sử dụng để xác định khi nào thì tăng/giảm mức ưu tiên của một tiến trình
* phương pháp được sử dụng để xác định queue nào mà tiến trình sẽ đến khi nó cần được phục vụ.

VD: **Ví dụ Multilevel Feedback Queue**

* Ba queue:
* *Q*0 – thời gian định lượng 8 ms
* *Q*1 – thời gian định lượng 16 ms
* *Q*2 – FCFS
* Lập Lịch:
* Một tiến trình vào queue *Q0* và được phục vụFCFS. Khi nó giành được CPU, tiến trình nhận được 8 ms. Nếu nó không hoàn thành trong 8 ms, tiến trình được chuyển tới queue *Q*1.
* Tại *Q*1 tiến trình tiếp tục được phục vụ FCFS với 16 ms nữa. Nếu nó vẫn chưa hoàn thành thì nó được ưu tiên và được chuyển đến queue *Q*2.
* **Multilevel Feedback Queues**

Diagram

Description automatically generated

Hình 12: Multilevel Feedback Queues

# **CHƯƠNG IV: CÁC LOẠI LẬP LỊCH**

1. **Lập lịch multiprocessor**

* Lập lịch CPU khi có nhiều processor phức tạp hơn nhiều
* Các loại processortrong multiprocessor
* *Đồng nhất (Homogeneous):* tất cả có cùng kiến trúc.
* *Không đồng nhất (Heterogeneous)*: một số tiến trình có thể không tương thích với kiến trúc của các CPU.
* *Cân bằng tải (Load balancing/sharing)*: một ready queue cho tất cả các processor, CPU nhàn rỗi được gán cho tiến trình ở đầu queue.
* *Đa xử lý không đối xứng - Asymmetric multiprocessing:*
* chỉ một processor (master processor) truy nhập các cấu trúc dữ liệu hệ thống, làm giảm sự cần thiết bảo vệ dữ liệu chia sẻ.

1. **Lập lịch thời gian thực**

* *Hard real-time* systems – yêu cầu hoàn thành một tác vụ găng (critical task) trong thời gian được đảm bảo.
* **resource reservation**: khi tiến trình được gửi đến cùng với lệnh cho biết thời gian cần thiết của nó, trình lập lịch có thể chấp nhận và đảm bảo nó sẽ kết thúc đúng hạn, hoặc từ chối tiến trình.
* *Soft real-time* computing – yêu cầu các tiến trình găng nhận mức ưu tiên trên các tiến trình kém may mắn hơn
* có thể phân phối tài nguyên không hợp lý, thời gian trễ lâu, starvation.
* phải cẩn thận trong thiết kế trình lập lịch và các khía cạnh liên quan của HĐH:
* lập lịch có ưu tiên, các tiến trình thời gian thực có mức ưu tiên cao nhất.
* trễ điều vận (dispatch latency) phải nhỏ.

1. **Lập lịch luồng**

* Lập lịch cục bộ (Local Scheduling):
* Bằng cách nào Thư viện luồng quyết định chọn luồng nào để đặt vào một CPU ảo khả dụng: Thường chọn luồng có mức ưu tiên cao nhất
* Sự cạnh tranh CPU diễn ra giữa các luồng của cùng một tiến trình.
* Trong các HĐH sử dụng mô hình Many-to-one, Many-to-many.
* Lập lịch toàn cục (Global Scheduling)
* Bằng cách nào kernel quyết định kernel thread nào để lập lịch CPU chạy tiếp.
* Sự cạnh tranh CPU diễn ra giữa tất cả các luồng trong hệ thống.
* Trong các HĐH sử dụng mô hình One-to-one (Windows XP, Linux, Solaris 9)

# **CHƯƠNG V: VÍ DỤ LẬP LỊCH TRONG HĐH LINUX**

1. **Linux scheduling**

* Linux kernel từ phiên bản 2.5:
* Cung cấp giải thuật lập lịch chạy trong thời gian hằng – *O* (1) – bất kể số lượng luồng trong hệ thống.
* Hỗ trợ tốt hơn cho các hệ thống đa xử lý đối xứng (SMP)
* Cung cấp sự công bằng và hỗ trợ cho các tác vụ (task) tương tác
* Trình lập lịch Linux dùng giải thuật có ưu tiên trước dựa trên mức ưu tiên với 2 dải riêng:
* Dải *real-time*: có mức từ 0 đến 99
* Dải *nice*: có mức từ 100 đến 140.
* Số càng nhỏ, mức ưu tiên càng cao
* Không giống với nhiều HĐH khác, Linux gán các task có mức ưu tiên càng cao càng được nhiều lượng thời gian q.

1. **Quan hệ giữa Priorities và Time-slice length**

Chart

Description automatically generated

Hình 13: Quan hệ giữa Priorities và Time-slice length

1. **Linux Scheduling (tiếp)**

* Kernel duy trì danh sách các task sẵn sàng trong một **runqueue**
* Nếu có nhiều bộ xử lý, mỗi BXL có 1 runqueue và lập lịch độc lập
* Mỗi runqueue có 2 mảng ưu tiên:
* *active*: chứa tất cả task vẫn còn trong lượng thời gian của chúng
* *expired*: chứa tất cả task đã hết lượng thời gian
* Trình lập lịch chọn task có mức ưu tiên cao nhất trong mảng active. Khi mảng active rỗng, 2 mảng sẽ tráo đổi cho nhau.
* Các real-time task được gán mức ưu tiên tĩnh
* Các task khác có mức ưu tiên động: = nice value ± 5
* Task có thời gian ngủ đợi vào/ra dài hơn thường là tương tác hơn
* tăng mức ưu tiên bằng cách -5
* Task có thời gian ngủ ngắn hơn thường là CPU-bound
* giảm mức ưu tiên bằng cách +5

# **CHƯƠNG VI: LỰA CHỌN GIẢI THUẬT**

* Chọn giải thuật lập lịch CPU nào cho hệ thống cụ thể?
* Trước tiên, xác định sử dụng tiêu chuẩn nào? Vd:
* Tối đa CPU utilization với ràng buộc response time lớn nhất là 1s
* Tối đa throughput để turnaround time là tỷ lệ tuyến tính với thời gian thực hiện

1. **Phân tích hiệu năng của từng giải thuật đối với các tiến trình**
2. **Sử dụng chuẩn hàng đợi: công thức Little: n = λ x W**

* n: độ dài queue trung bình
* W: thời gian chờ đợi trung bình trong queue
* **λ**: tốc độ đến queue của tiến trình (số tiến trình/giây)

1. **Mô phỏng: lập trình mô hình hệ thống để đánh giá**
2. **Thực hiện: đặt giải thuật cụ thể trong hệ thống thực để đánh giá**

**CHƯƠNG VII: Mô phỏng và đánh giá một số giải thuật Scheduling**

1. **Một số thông số cần lưu ý**

* *Arrival Time (AT)*: Thời gian Process đến hàng đợi Ready Queue.
* *Completion Time (CT):* Thời gian Process hoàn thành quá trình thực thi.
* *Burst Time (BT):* Thời gian cần thiết để CPU thực thi tiến trình
* *Turn Around Time (TAT):* Thời gian chênh lệch giữa *Completion Time* và *Arrival Time***.**

Turn Around Time = Completion Time – Arrival Time

* *Waiting Time (WT):* Thời gian chênh lệch giữa *Turn Around Time* và *Burst Time.*

Waiting Time = Turn Around Time – Burst Time

1. **Mô phỏng một số giải thuật**

* Xây dựng chương trình mô phỏng đưa vào các thông số như: AT, BT để tính ra các thông số cần thiết còn lại.
* Code: github.com/duongkien27/[OS/process\_scheduling\_algo.c](https://github.com/duongkien27/OS/blob/main/process_scheduling_algo.c)
* Các thuật toán tham khảo thuật toán trên Geeksforgeeks

Text

Description automatically generated

Hình 14: Mô phỏng giải thuật

### **2.1. First Come – First Served (FCFS)**

* Giả lập có 4 Process với AT và BT theo thứ tự: {0,2,4,5} và {7,4,1,4} (ms)
* Kết quả:

Text

Description automatically generated

Hình 15: First come - first serve

* **Nhận xét:** FIFO “không cồng bằng” với các process có CPU burst ngắn vì nó phải chờ trong thời gian dài (so với thời gian mà nó cần phục vụ) mới được sử dụng CPU. FIFO sẽ ưu tiên các process thuộc dạng CPU bound.

### **2.2. Round – Robin (RR)**

* Giả lập có 4 Process với AT và BT theo thứ tự: {0,1,2,3} và {7,2,1,2} (ms)
* Kết quả:
* QT = 2 ms

Text

Description automatically generated with medium confidence

Hình 16: RR Qt = 2ms

* QT = 20 ms

Text

Description automatically generated with low confidence

Hình 17: RR Qt = 20ms

* *Nhận xét***:** Khi QT ngắn, thì đáp ứng nhanh do có nhiều lần *process switch*

Khi QT dài hơn thì throughtput tốt hơn nhưng đáp ứng chậm hơn. Time Slice quá lớn, RR sẽ trở thành FCFS.

### **2.3. Short Job First (SJF)**

* Giả lập có 4 Process với AT và BT theo thứ tự: {0,2,4,5} và {7,4,1,4} (ms)
* Kết quả:

Text

Description automatically generated with medium confidence

Hình 18: SJF

* *Nhận xét:*SJF sử dụng ưu tiên nhầm định: công việc ngắn nhất được ưu tiên trước. Process có thời gian thực thi dài có thể bị trì hoãn vô thời hạn.

Không thích hợp với môi trường time-sharing khi không dung Preemtive.

# **CHƯƠNG VIII: ĐÁNH GIÁ TRÊN LINUX**

1. **Bài toán đặt ra:**

* Khởi tạo**:** Tạo 3 tiến trình P1, P2, P3 chạy liên tục với vòng lặp vô hạn.
* Theo dõi**:** Theo dõi mức độ sử dụng tài nguyên của P1, P2, P3 trong 5 phút ứng với các trường hợp sau:
* *Trường hợp 1:* Scheduling Policies: SCHED\_OTHER, thay đổi NI của các tiến trình.
* *Trường hợp 2:* Scheduling Policies: P1- SCHED\_OTHER; P2- SCHED\_FIFO; P3- SCHED\_RR. NI của P1, P2, P3 đều bằng 0.
* *Trường hợp 3:* Scheduling Policies tương tự trường hợp trên nhưng thay đổi NI value của tiến trình P1.

1. **Trường hợp 1: SCHED\_OTHER; NI thay đổi**

* Mức độ ưu tiên của tiến trình: Được thể hiện bởi priority của tiến trình, hay giá trị NI (nice value) của tiến trình đó.
* Nice value thay đổi trong khoảng -20 đến 19.
* Giá trị NI càng thấp thì mức độ ưu tiên càng cao.
* Để thay đổi giá trị NI xuống mức âm (nâng cao mức độ ưu tiên) cần phải có quyền root thông qua lệnh sudo.
* Xây dựng chương trình renice.c ([github.com/duongkien27/OS/renice.c](https://github.com/duongkien27/OS/blob/main/renice.c)) để thực hiện thay đổi priority của tiến trình và theo dõi kết quả.
* Theo dõi:Thời gian sử dụng tài nguyên của tiến trình P1, P2, P3 trong 5 phút.
* Kết quả:

Table

Description automatically generated

Hình 19: thời gian sử dụng tài nguyên

Hình 20: Thời gian sử dụng tài nguyên trong 5 phút của tiến trình

* *Kết luận***:** Thời gian chiếm dụng tài nguyên của 3 tiến trình P1, P2, P3 tương đương nhau khi ở SCHED\_OTHER và cùng nice value.

1. **Trường hợp 2: Thay đổi Scheduling Policies**

* Cho 3 tiến trình P1, P2, P3 chạy đồng thời.
* Đặt tiến trình P1 ở SCHED\_OTHER.
* Đặt tiến trình P2 ở SCHED\_FIFO.
* Đặt tiến trình P3 ở SCHED\_RR.
* Theo dõi thời gian sử dụng tài nguyên của 3 tiến trình P1, P2, P3 trong 5 phút.
* Kết quả:

Text

Description automatically generated with medium confidence

Hình 21: Thời gian sử dụng tài nguyên

Hình 22:Thời gian sử dụng CPU theo SCHED

* *Kết luận***:** Thời gian sử dụng tài nguyên của tiến trình P1 ở SCHED\_OTHER thấp hơn thời gian sử dụng tài nguyên của P2 (SCHED\_FIFO) và P3 (SCHED\_RR)

1. **Trường hợp 3: Tương tự TH2, thay đổi NI**

* Cho 3 tiến trình P1, P2, P3 chạy đồng thời.
* Đặt tiến trình P1 ở SCHED\_OTHER.
* Đặt tiến trình P2 ở SCHED\_FIFO.
* Đặt tiến trình P3 ở SCHED\_RR.
* Thay đổi NI của tiến trình P1 xuống -15 (mức độ ưu tiên cao)
* Theo dõi thời gian sử dụng tài nguyên của 3 tiến trình P1, P2, P3 trong 5 phút.
* Kết quả:

Text

Description automatically generated with medium confidence

Hình 23: thời gian sử dụng tài nguyên

Hình 24: thời gian sử dụng CPU theo NI, SCHED

* *Kết luận:*
* Thời gian sử dụng tài nguyên của tiến trình P1 ở SCHED\_OTHER thấp hơn thời gian sử dụng tài nguyên của P2 (SCHED\_FIFO) và P3 (SCHED\_RR)
* Từ kết quả *trường hợp 2* và *trường hợp 3* có thể thấy:

∑ t (SCHED\_FIFO) = ∑ t (SCHED\_RR)

Và không bị ảnh hưởng bởi nice value

1. **Theo dõi sự khác biệt trên Web Server:**

* Sử dụng dịch vụ EC2 – Virtual Machine của AWS Amazon để thuê Server VPS Ubuntu trên cụm máy chủ N. Virginia

Graphical user interface, application

Description automatically generated

Hình 25: Theo dõi sự khác biệt trên Webserver

* Cài đặt môi trường LAMP (Linux – Apache – MySQL - PHP) để tạo thành Server Web.
* Thay đổi Scheduler cho chương trình ./testserver và theo dõi thời gian phản hồi của Website (trung bình cho 5 lần đo)

Table

Description automatically generated

Hình 26: Theo dõi thời gian phản hồi của Webserver

1. **Nhận xét các kết quả tổng quan:**

* Các số liệu chúng em đo được chưa hoàn toàn chính xác, chỉ dừng lại ở mức độ có thể quan sát được bằng đồng hồ bấm giờ
* Chưa đo được cái số liệu chi tiết trong các trường hợp Scheduling (ex: time quantum, …)
* Các số liệu đo trong trường hợp khan hiếm tài nguyên chỉ dừng lại ở mức độ Server nhận 10 request cùng lúc bằng cách thủ công (truy cập từ nhiều nơi cùng lúc)

# **CHƯƠNG IX: TỔNG KẾT**

Thông qua bài tập lớn bộ môn Hệ điều hành dưới sự giảng dậy của TS. Hàn Huy Dũng, nhóm 11 chúng em đã tích lũy được rất nhiều kiến thức bổ ích như: Hiểu về tầm quan trọng của việc Scheduling, các tiêu chí Scheduling, các giải thuật của Scheduling, các kiến thức liên quan như thay đổi mức ưu tiên của tiến trình, đánh giá kết quả, … Đó đều là những kết thức bổ ích, những hành trang để chúng em tìm hiểu sâu hơn nữa vào hệ điều hành.

Chúng em cảm ơn thầy và các bạn đã đồng hành cùng môn học suốt môn kỳ học vừa qua. Mong thầy và các bạn có những đóng góp sửa đổi để bài báo cáo của chúng em có thể tiến bộ và hoàn thiện hơn nữa

Chúng em xin cảm ơn!

**TÀI LIỆU THAM KHẢO**

<https://man7.org/linux/man-pages/man7/sched.7.html>

<https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/>

<https://www.youtube.com/watch?v=EWkQl0n0w5M&list=PLBlnK6fEyqRitWSE_AyyySWfhRgyA-rHk>