Lab2-answer-2023-2024

Kiến trúc máy tính và Hệ điều hành (Đại học Kinh tế Quốc dân)

OS - Class question 2

Student's fullname: Nguyễn Thạc Hoàng Nam *Student's ID:11224467*

1. **Kể tên 1 hệ thống chỉ cho phép một tiến trình thực thi tại một thời điểm thời gian? 1 hệ thống cho phép nhiều tiến trình thực thi đồng thời?**
   - Hệ thống chỉ cho phép 1 tiến trình thực thi tại 1 thời điểm là quản lí tiến trình
   - Hệ thống cho phép nhiều tiến trình thực thi đồng thời là quản lí bộ nhớ chính
2. **Trong Unix, lời gọi hệ thống nào được sử dụng để tạo tiến trình.**
   - fork()
3. **Một tiến trình sẽ bị kết thúc trong các trường hợp nào?**
   - ❖ Tiến trình thực thi câu lệnh cuối cùng và hỏi hệ điều hành để ra quyết định cho nó.**(exit)**
     - ➢ Đưa ra dữ liệu từ con tới cha (thông qua cơ chế **wait**)
     - ➢ Loại bỏ PCB của tiến trình
     - ➢ Loại bỏ tiến trình ra khỏi toàn bộ các danh sách quản lý hệ thống
     - ➢ Tài nguyên của tiến trình bị thu hồi bởi hệ điều hành (**Tái cấp phát**)
   - ❖ Tiến trình cha có thể chấm dứt việc thực thi của tiến trình con **(abort)**
     - ➢ Tiến trình con đã vượt quá số tài nguyên được cấp phát
     - ➢ Nhiệm vũ được giao cho tiến trình con không còn cần thiết nữa
     - ➢ Tiến trình cha kết thúc, hệ điều hành không cho phép tiến trình con tiếp tục nữa
4. **Trạng thái Sẵn sàng của tiến trình là gì?**
   - Là trạng thái tiến trình chờ được cấp phát CPU để thực thi
5. **Các tiến trình liên lạc với nhau để làm gì?**
   - ❖ Các tiến trình liên lạc với nhau để:
     - ➢ Chia sẻ thông tin
     - ➢ Tăng tốc tính toán
     - ➢ Tính mô đun
     - ➢ Sự tiện lợi
6. **Khi mở một tab mới trên trình duyệt chrome, trình duyệt này có tạo ra tiến trình mới không?**
   - Có, khi mở một tab mới trên trình duyệt Chrome, trình duyệt này thường tạo ra một tiến trình mới cho tab đó.
7. **Tiến trình và luồng khác nhau gì?**
   - Một tiến trình là một chương trình đang chạy với không gian bộ nhớ riêng, trong khi một luồng là một phần của một tiến trình, chia sẻ không gian bộ nhớ và các tài nguyên khác với các luồng khác trong cùng một tiến trình.

Process	Thread
Process is considered heavy weight	Thread is considered light weight
Unit of Resource Allocation and of protection	Unit of CPU utilization
Process creation is very costly in terms of resources	Thread creation is very economical
Program executing as process are relatively slow	Programs executing using thread are comparatively faster
Process cannot access the memory area belonging to another process	Thread can access the memory area belonging to another thread within the same process
Process switching is time consuming	Thread switching is faster
One Process can contain several threads	One thread can belong to exactly one process

8. **Tại sao các hệ điều hành hiện đại hỗ trợ môi trường đa nhiệm ?**

- Vì các hệ điều hành hỗ trợ môi trường đa nhiệm mang lại nhiều lợi ích quan trọng cho người người sử dụng như tối đa hóa hiệu suất sử dụng tài nguyên, chia sẻ thông tin và tài nguyên, phản hồi nhanh chóng và hỗ trợ các ứng dụng chạy đa nhiệm

9. Sự khác biệt, mối quan hệ giữa tiến trình(Process) và tiểu trình(Thread) ?

- Tiến trình là một bộ phận của chương trình đang thực hiện. Tiến trình là đơn vị làm việc cơ bản của hệ thống, trong hệ thống có thể tồn tại nhiều tiến trình cùng hoạt động, trong đó có cả tiến trình của hệ điều hành và tiến trình của chương trình người sử dụng. Các tiến trình này có thể hoạt động đồng thời với nhau.
- Tiểu trình cũng là đơn vị xử lý cơ bản trong hệ thống, nó cũng xử lý tuần tự đoạn code của nó, nó cũng sở hữu một con trỏ lệnh, một tập các thanh ghi và một vùng nhớ stack riêng và các tiểu trình cũng chia sẻ thời gian xử lý của processor như các tiến trình.
- Trong một tiến trình có thể có nhiều tiểu trình. Các tiểu trình trong một tiến trình chia sẻ một không gian địa chỉ chung, điều này có nghĩa các tiểu trình có thể chia sẻ các biến toàn cục của tiến trình, có thể truy xuất đến stack của tiểu trình khác trong cùng tiến trình.

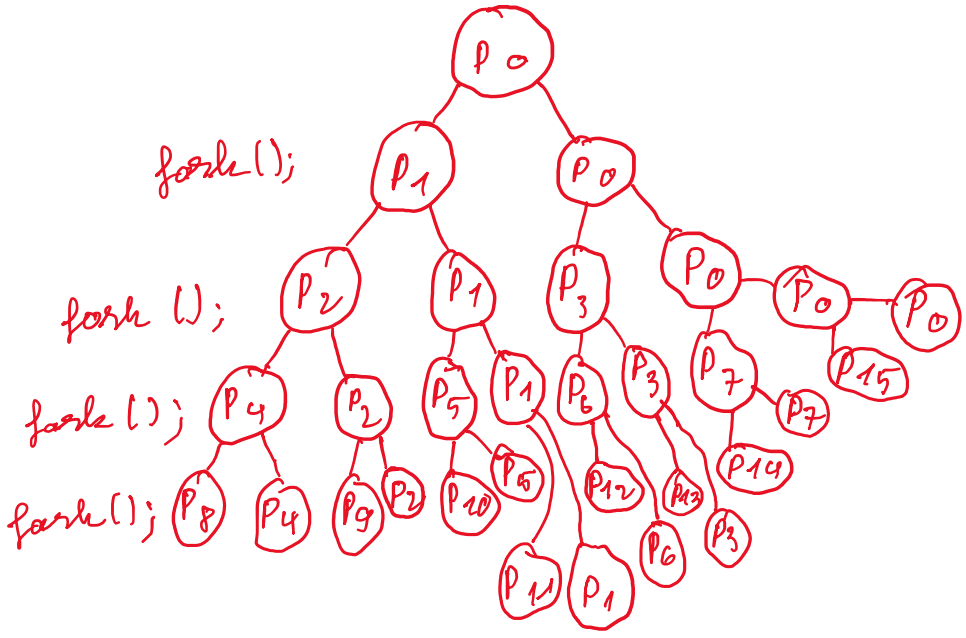
10. Tìm một số ứng dụng thích hợp với mô hình đa tiến trình; và một số ứng dụng thích hợp với mô hình đa tiểu trình.

- Ứng dụng thích hợp với mô hình đa tiến trình:
  - Trình duyệt web: Google Chrome
  - Hệ thống máy chủ: máy chủ web, máy chủ dịch vụ thư điện tử, thanh toán
  - Trình sử lý văn bản đa nhiệm: MS word

11. Lệnh sau tạo ra bao nhiêu tiến trình. Hãy vẽ cây tiến trình tạo ra bởi chương trình sau

```
include <stdio.h>
include <unistd.h>
int main()
{
        fork();wait(NULL)
        fork();wait(NULL)
        fork();wait(NULL)
        fork();wait(NULL)
        return 0;
}
```

- Số tiến trình tạo ra = 2^n; n là số lần gọi câu lệnh fork(); → có 16 tiến trình
-



12. Cho chương trình sau, xác định giá trị của pid tại các dòng A, B, C, và D. Giả sử rằng pid của tiến trình cha là 1500, pid của tiến trình con là 1506

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid, pid1;

//fork a child process
pid = fork();

if (pid< 0) //error occurred
{
        fprintf(stderr, "Fork Failed");
        return 1;
}
else if (pid == 0) //child process
{
        pid1 = getpid();
        printf("child: pid = %d", pid); //line A
        printf("child: pid1 = %d", pid1); //line B
}
else //parent process
{
        pid1 = getpid();
        printf("parent: pid = %d", pid); //line C
        printf("parent: pid1 = %d", pid1); //line D
        wait(NULL);
}

return 0;
}
```

- Line A: pid = 0
- Line B: pid = 1506
- Line C: pid = 1506
- Line D: pid = 1500

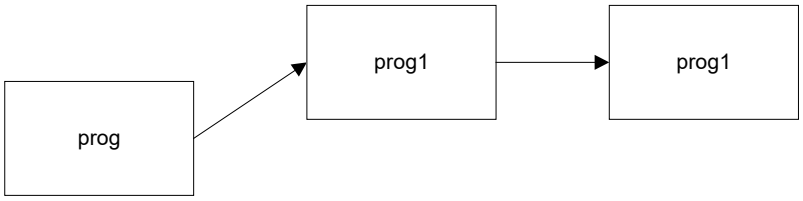
13. Xác định giá trị được in ra tại dòng A

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;

int main()
{
        pid_t pid;
        pid = fork();

        if (pid == 0) //child process
        {
                value += 15;
                return 0;
        }
        else (pid > 0) //parent process
        {
                wait(NULL);
                printf("Parent: value = %d", value); //Line A
                return 0;
        }
}
```

- Parent: value = 5

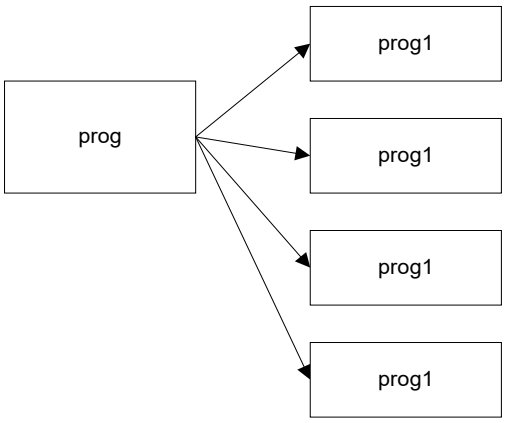
14. Viết chương trình “prog” tạo ra các cây tiến trình sau

a) Cây tiến trình A




```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
        pid_t pid;
        pid = fork(); //fork a child

        if (pid == 0) //child process
        {
                pid_t pid1;
                pid1 = fork();
        }
        else if (pid < 0)
        {
                //error
        }
        else //parent process
        {
                wait(NULL);
        }
}
```

b) Cây tiến trình B




```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
	pid_t pid1, pid2, pid3, pid4;

	pid1 = fork();
	if(pid1 < 0)
	{
		//error
	}
	else if(pid1 == 0)
	{
		// child process
	}
	else
	{
		//parent process
	}

	Pid2 = fork();
	if(pid1 < 0)
	{
		//error
	}
	else if(pid1 == 0)
	{
		// child process
	}
	else
	{
		//parent process
	}

	Pid3 = fork();
```

```
        if(pid1 < 0)
        {
                //error
        }
        else if(pid1 == 0)
        {
                // child process
        }
        else
        {
                //parent process
        }

        Pid4 = fork();
        if(pid1 < 0)
        {
                //error
        }
        else if(pid1 == 0)
        {
                // child process
        }
        else
        {
                //parent process
        }

}
```

Lệnh nào trong các lệnh sau nên là lệnh đặc quyền?

Đặt giá trị cho timer

Đọc đồng hồ

Xóa bộ nhớ

Đưa ra lệnh tạo bẫy

Tắt các ngắt

Thay đổi các mục trong bảng trạng thái thiết bị

Chuyển từ chế độ người dùng sang chế độ hạt nhân

Truy nhập thiết bị I/O

15. Xét tập các tiến trình sau (với thời gian yêu cầu CPU và độ ưu tiên kèm theo) :

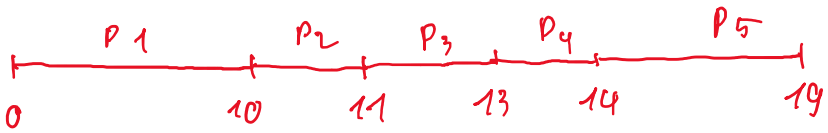
Tiến trình	Thời điểm vào RL	Thời gian CPU	Độ ưu tiên
P_1	0	10	3
P_2	1	1	1
P_3	2	2	3
P_4	3	1	4
P_5	4	5	2

Giả sử các tiến trình cùng được đưa vào hệ thống tại thời điểm 0

a) **Cho biết kết quả điều phối hoạt động của các tiến trình trên theo thuật toán FIFO; SJF; điều phối theo độ ưu tiên độc quyền (độ ưu tiên 1 > 2 > ...); và RR (quantum=2).**
b) **Cho biết thời gian lưu lại trong hệ thống (turnaround time) của từng tiến trình trong từng thuật toán điều phối ở câu a.**
c) **Cho biết thời gian chờ trong hệ thống (waiting time) của từng tiến trình trong từng thuật toán điều phối ở câu a.**
d) **Thuật toán điều phối nào trong các thuật toán ở câu a cho thời gian chờ trung bình là cực tiểu ?**

- FIFO:

Gantt chart



Waiting time : $P_1 : 0$
$P_2 : 10 - 1 = 9$
$P_3 : 11 - 2 = 9$
$P_4 : 13 - 3 = 10$
$P_5 : 14 - 4 = 10$

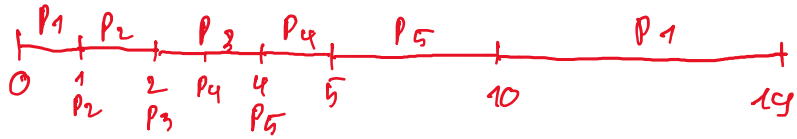
Turnaround time : $P_1 : 0 + 10 = 10$
$P_2 : 9 + 1 = 10$
$P_3 : 9 + 2 = 11$
$P_4 : 10 + 1 = 11$
$P_5 : 10 + 5 = 15$

Average waiting time $= \frac{0+9+9+10+10}{5} = 7,6$

Average Turnaround time $= 7,6 + \frac{19}{5} = 11,4$

- **SJF**
- **preemptive**

Gantt chart



Waiting time:

$P_1: 0 + 10 - 1 = 9$

$P_2: 1 - 1 = 0$

$P_3: 2 - 2 = 0$

$P_4: 4 - 3 = 1$

$P_5: 5 - 4 = 1$

Turnaround time:

$P_1: 9 + 10 = 19$

$P_2: 0 + 1 = 1$

$P_3: 0 + 2 = 2$

$P_4: 1 + 1 = 2$

$P_5: 1 + 5 = 6$

Average waiting time:

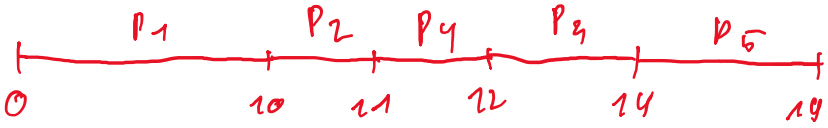
$$\frac{9 + 0 + 0 + 1 + 1}{5} = \frac{11}{5} = 2.2$$

Average turnaround time:

$$\frac{19 + 1 + 2 + 2 + 6}{5} = 6$$

Non-preemtive

Gantt chart



Waiting time: $P_1: 0$

$P_2: 10 - 1 = 9$

$P_3: 12 - 2 = 10$

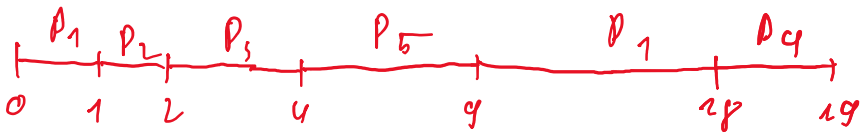
$P_4: 11 - 3 = 8$

$P_5: 14 - 4 = 10$

Average waiting time: $\frac{0 + 9 + 10 + 8 + 10}{5} = \frac{37}{5} = 7.4(8)$

- **Điều phối theo độ ưu tiên độc quyền**

Grant chart



Waiting time $P_1 : 0 + 9 - 1 = 8$

$P_2 : 1 - 1 = 0$

$P_3 : 2 - 2 = 0$

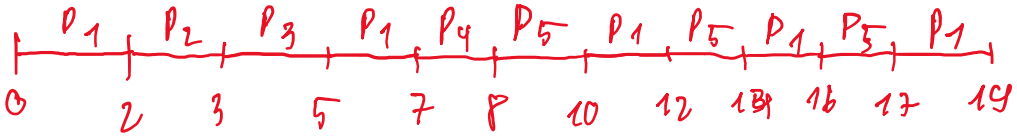
$P_4 : 18 - 3 = 15$

$P_5 : 4 - 4 = 0$

Avg waiting time $= \frac{8 + 15}{5} = \frac{23}{5} = 4,6 (s)$

- **Round robin (quantum = 2)**

Grant chart



Waiting time : $P_1 : 0 + 5 - 2 + 10 - 7 + 14 - 12 + 17 - 16 = 9$

$P_2 : 2 - 1 = 1$

$P_3 : 3 - 2 = 1$

$P_4 : 7 - 3 = 4$

$P_5 : 8 - 4 + 12 - 10 + 16 - 14 = 8$

Average waiting time $= \frac{9 + 1 + 1 + 4 + 8}{5} = 4,6 (s)$

⇨ Trong những thuật toán trên, thuật toán Round Robin (q = 2) có thời gian chờ trung bình là cực tiểu.

16. Giả sử có các tiến trình sau trong hệ thống :

Tiến trình	Thời điểm vào RL	Thời gian CPU
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

Sử dụng nguyên tắc điều phối độc quyền và các thông tin có được tại thời điểm ra quyết định để trả lời các câu hỏi sau đây :

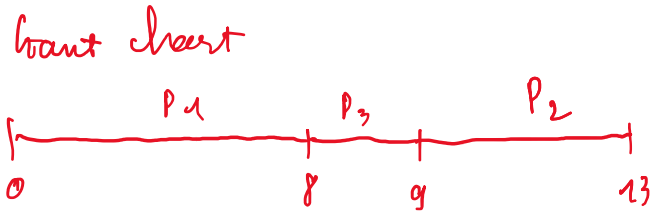
a) Cho biết thời gian lưu lại trung bình trong hệ thống (turnaround time) của các tiến trình trong thuật toán điều phối FIFO.

Waiting time : $P_1 = 0$; $P_2 = 8 - 0,4 = 7,6$; $P_3 = 12 - 1 = 11$

Average waiting time : $\frac{0 + 7,6 + 11}{3} = 6,2(s)$

Average turnaround time = $6,2 + \frac{13}{3} = 10,53(s)$

b) Cho biết thời gian lưu lại trung bình trong hệ thống (turnaround time) của các tiến trình trong thuật toán điều phối SJF.

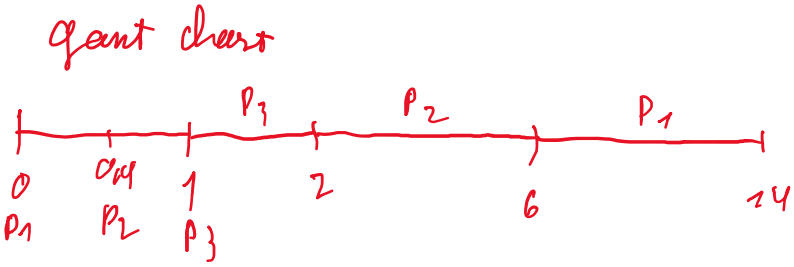


Waiting time : $P_1 = 0$; $P_2 = 9 - 0,4 = 8,6$; $P_3 = 8 - 1 = 7$

Avg waiting time = $\frac{8,6 + 7}{3} = 5,2(s)$

Avg turnaround time = $5,2 + \frac{13}{3} = 9,53(s)$

c) **Thuật toán SJF dự định cải tiến sự thực hiện của hệ thống , nhưng lưu ý chúng ta phải chọn điều phối P1 tại thời điểm 0 vì không biết rằng sẽ có hai tiến trình ngắn hơn vào hệ thống sau đó . Thử tính thời gian lưu lại trung bình trong hệ thống nếu để CPU nhàn rỗi trong 1 đơn vị thời gian đầu tiên và sau đó sử dụng SJF để điều phối. Lưu ý P1 và P2 sẽ phải chờ trong suốt thời gian nhàn rỗi này, do vậy thời gian chờ của chúng tăng lên. Thuật toán điều phối này được biết đến như điều phối dựa trên thông tin về tương lai.**



Waiting time $P_1: 6-0=6$ $P_3: 1-1=0$

$P_2: 2-0,4=1,6$

Avg waiting time $= \frac{6+0+1,6}{3} = \frac{7,6}{3} = 2,53$ (s)

Avg turnaround time $= 2,53 + \frac{18}{3} = 8,53$ (s)

17. **Phân biệt sự khác nhau trong cách tiếp cận để ưu tiên cho tiến trình ngắn trong các thuật toán điều phối sau :**

a) **FIFO.**

Trong thuật toán này, độ ưu tiên phục vụ tiến trình căn cứ vào thời điểm hình thành tiến trình. Hàng đợi các tiến trình được tổ chức theo kiểu FIFO. Mọi tiến trình đều được phục vụ theo trình tự xuất hiện cho đến khi kết thúc hoặc bị ngắt.

b) **RR**

Giải thuật định thời luân phiên (round-robin scheduling algorithm-RR) được thiết kế đặc biệt cho hệ thống chia sẻ thời gian. Tương tự như định thời FIFO nhưng sự trưng dụng CPU được thêm vào để chuyển CPU giữa các quá trình. Đơn vị thời gian nhỏ được gọi là định mức thời gian (time quantum) hay phần thời gian (time slice) được định nghĩa. Định mức thời gian thường từ 10 đến 100 mili giây. Hàng đợi sẵn sàng được xem như một hàng đợi vòng. Bộ định thời CPU di chuyển vòng quanh hàng đợi sẵn sàng, cấp phát CPU tới mỗi quá trình có khoảng thời gian tối đa bằng một định mức thời gian. Để cài đặt định thời RR, chúng ta quản lý hàng đợi sẵn sàng như một hàng đợi FIFO của các quá trình. Các quá trình mới được thêm vào đuôi hàng đợi. Bộ định thời CPU chọn quá trình đầu tiên từ hàng đợi sẵn sàng, đặt bộ đếm thời gian để ngắt sau 1 định mức thời gian và gởi tới quá trình. Sau đó, một trong hai trường hợp sẽ xảy ra. Quá trình có 1chu kỳ CPU ít hơn 1 định mức thời gian. Trong trường hợp này, quá trình sẽ

tự giải phóng. Sau đó, bộ định thời biểu sẽ xử lý quá trình tiếp theo trong hàng đợi sẵn sàng. Ngược lại, nếu chu kỳ CPU của quá trình đang chạy dài hơn 1 định mức thời gian thì độ đếm thời gian sẽ báo và gây ra một ngắt tới hệ điều hành. Chuyển đổi ngữ cảnh sẽ được thực thi và quá trình được đặt trở lại tại đuôi của hàng đợi sẵn sàng. Sau đó, bộ định thời biểu CPU sẽ chọn quá trình tiếp theo trong hàng đợi sẵn sàng.

c) Điều phối với độ ưu tiên đa cấp

Mỗi tiến trình được gán cho một độ ưu tiên tương ứng, tiến trình có độ ưu tiên cao nhất sẽ được chọn để cấp phát CPU đầu tiên. Độ ưu tiên có thể được định nghĩa nội tại hay nhờ vào các yếu tố bên ngoài. Độ ưu tiên nội tại sử dụng các đại lượng có thể đo lường để tính toán độ ưu tiên của tiến trình, ví dụ các giới hạn thời gian, nhu cầu bộ nhớ…Độ ưu tiên cũng có thể được gán từ bên ngoài dựa vào các tiêu chuẩn do hệ điều hành như tầm quan trọng của tiến trình, loại người sử dụng sỡ hữu tiến trình…

Giải thuật điều phối với độ ưu tiên có thể theo nguyên tắc độc quyền hay không độc quyền. Khi một tiến trình được đưa vào danh sách các tiến trình sẵn sàng, độ ưu tiên của nó được so sánh với độ ưu tiên của tiến trình hiện hành đang xử lý. Giải thuật điều phối với độ ưu tiên và không độc quyền sẽ thu hồi CPU từ tiến trình hiện hành để cấp phát cho tiến trình mới nếu độ ưu tiên của tiến trình này cao hơn tiến trình hiện hành. Một giải thuật độc quyền sẽ chỉ đơn giản chèn tiến trình mới vào danh sách sẵn sàng, và tiến trình hiện hành vẫn tiếp tục xử lý hết thời gian dành cho nó.

18. Giả sử một hệ điều hành áp dụng giải thuật điều phối multilevel feedback với 5 mức ưu tiên (giảm dần). Thời lượng quantum dành cho hàng đợi cấp 1 là 0,5s. Mỗi hàng đợi cấp thấp hơn sẽ có thời lượng quantum dài gấp đôi hàng đợi ứng với mức ưu tiên cao hơn nó. Một tiến trình khi vào hệ thống sẽ được đưa vào hàng đợi mức cao nhất, và chuyển dần xuống các hàng đợi bên dưới sau mỗi lượt sử dụng CPU. Một tiến trình chỉ có thể bị thu hồi CPU khi đã sử dụng hết thời lượng quantum dành cho nó. Hệ thống có thể thực hiện các tác vụ xử lý theo lô hoặc tương tác, và mỗi tác vụ lại có thể hướng xử lý hay hướng nhập xuất.

a) Giải thích tại sao hệ thống này hoạt động không hiệu quả ?

- Hệ thống có quantum quá ngắn dẫn đến phân mảnh tiến trình quá nhiều.

b) Cần phải thay đổi (tối thiểu) như thế nào để hệ thống điều phối các tác vụ với những bản chất khác biệt như thế tốt hơn ?

- Cần tăng quantum lên mức phù hợp.

19. Phân biệt Concurrency (đồng thời) và parallelism (song song).

- Xử lý concurrency (đồng thời) nghĩa là có khả năng giải quyết nhiều công việc một lúc, và những công việc đó ko nhất thiết phải xảy ra tại cùng một thời điểm.
- Parallelism (song song) nghĩa là có khả năng xử lý nhiều công việc tại cùng một thời điểm.

20. Xem xét giải thuật Banker với ma trận yêu cầu cấp phát như sau:

Processes	Allocation A B C	Max A B C	Available A B C
P0	**1 1 2**	**4 3 3**	**2 1 0**
P1	**2 1 2**	**3 2 2**	
P2	**4 0 1**	**9 0 2**	
P3	**0 2 0**	**7 5 3**	
P4	**1 1 2**	**1 1 2**	

1. Tính ma trận Need?

AVAILABLE			NEED			P
A	B	C	A	B	C	
2	1	0	3	2	1	
4	2	2	1	1	0	P1
5	3	4	5	0	1	P4
6	4	6	7	3	3	P0
10	4	7	0	0	0	P2
10	6	7				P3

2. Kiểm tra xem hệ thống có ở trạng thái an toàn hay không?

- Hệ thống có ở trạng thái an toàn vì tồn tại thứ tự an toàn: P1, P4, P0, P2, P3

3. Xác định tổng tổng của từng loại tài nguyên?

A – 10 tài nguyên

B – 6 tài nguyên

C – 7 tài nguyên.