**Giải thuật đã lựa chọn: Least Laxity First (LLF)**

Thuật toán LLF (Least Laxity First) là một thuật toán điều phối **ưu tiên động**, nghĩa là mức độ ưu tiên của các tiến trình sẽ thay đổi theo thời gian thực. Tại mỗi thời điểm, hệ thống sẽ ưu tiên tiến trình có **độ linh động (laxity)** nhỏ nhất. Thuật toán này cho phép ưu tiên những tiến trình đang có nguy cơ trễ hạn, giúp cải thiện tỷ lệ hoàn thành đúng deadline.

### **1. Khái niệm độ linh động (Laxity):**

Laxity là khoảng thời gian mà một tiến trình có thể trì hoãn thực hiện mà vẫn hoàn thành đúng hạn.  
Laxity của một tiến trình tại thời điểm t được tính theo công thức:

*Laxity = Deadline − Current Time − Remaining Execution Time*

Trong đó:

*Deadline* là thời hạn hoàn thành tiến trình

*Current Time* là thời gian hiện tại (thời gian thực)

*Remaining Execution Time* là thời gian thực hiện của tiến trình

Laxity trả về một số lớn hơn hoặc bằng 0. Nếu giá trị nó nhỏ hơn 0, tức là chắc chắn tiến trình đó sẽ bị chậm deadline.

### **2. Nguyên tắc hoạt động của LLF:**

* Tại mỗi đơn vị thời gian, tính **laxity** cho tất cả các tiến trình đang sẵn sàng.
* Chọn tiến trình có **laxity nhỏ nhất** để thực hiện. Nếu CPU đang có tiến trình mà có tiến trình khác có laxity nhỏ hơn tiến trình trong CPU, CPU sẽ ưu tiên thực hiện tiến trình có laxity nhỏ hơn trước.
* Nếu có nhiều tiến trình có cùng laxity nhỏ nhất, chọn tiến trình đến trước.
* Cập nhật laxity của tất cả các tiến trình. Tiến trình nào có laxity nhỏ hơn 0 (chắc chắn chậm deadline),**bỏ qua nó và lập lịch cho các tiến trình khác.**

### **3. Triển khai, mô tả chương trình**

Trong chương trình, mỗi tiến trình được biểu diễn dưới dạng một cấu trúc Task, định nghĩa như sau: typedef struct {

* char id; // Tên tiến trình
* int exec\_time; // Thời gian thực thi
* int deadline; // Deadline tương đối
* int period; // Chu kỳ lặp lại
* int next\_release; // Thời điểm release tiếp theo
* int remaining; // Thoi gian con lai de chay
* int abs\_deadline; // Deadline tuyet doi cua lan hien tai
* int active; // Co ton tai hay khong (1: ton tai, 0: da xoa)

} Task;

- **int** laxity(**Task\*** task, **int** current\_time)

* Đầu vào: task: con trỏ tới một tiến trình; current\_time: thời điểm hiện tại.
* Đầu ra: Giá trị laxity (độ linh động): deadline - current\_time - remaining, dùng để chọn tiến trình ưu tiên nhất trong LLF.

- **void** llf\_scheduler(**Task** tasks[], **int** n)

* Hàm nhận vào một mảng các tiến trình (kiểu Task) gồm tối đa n phần tử.
* Chức năng: Hàm thực hiện lập lịch các tiến trình theo thuật toán LLF, trong đó ở mỗi đơn vị thời gian, tính toán laxity của từng tiến trình.
  + Tiến trình có độ lỏng thời hạn (laxity) nhỏ nhất sẽ được chọn để thực thi.
  + Hàm xử lý theo mô hình preemptive, cho phép chuyển CPU sang tiến trình khác nếu xuất hiện tiến trình có laxity thấp hơn. Nếu tại thời điểm hiện tại không có tiến trình khả dụng, hệ thống sẽ ở trạng thái nhàn rỗi.
  + Kết thúc quá trình, hàm sẽ in ra biểu đồ Gantt thể hiện lịch sử lập lịch.

- **int** main(): chứa danh sách khởi tạo các tiến trình và gọi hàm llf\_scheduler.

### **4. Kết quả minh họa:**

**A screenshot of a computer

AI-generated content may be incorrect.**

*Hình 2; Danh sách tiến trình đầu vào, số lượng tiến trình, gọi hàm main*

**A black background with white text

AI-generated content may be incorrect.**

*Hình 3; Danh sách tiến trình hoàn thành kịp deadline / không kịp deadline và biểu đồ Gantt*

### **Tài liệu tham khảo:**

<https://microcontrollerslab.com/least-laxity-first-llf/>

<https://www.amd.e-technik.uni-rostock.de/veroeff/scopell.pdf>

https://scispace.com/pdf/comparison-of-real-time-task-scheduling-algorithms-1ckegbamqo.pdf