MÔN: HỆ ĐIỀU HÀNH
Bài thực hành số 2.1: Multiprogramming

I. Mục tiêu

- Giúp SV làm quen với việc lập trình để kích hoạt/dừng process bất kỳ.

II. Nội dung

- Xây dựng chương trình nhỏ cho phép người dùng chọn file khả thi cần chạy, kích hoạt chạy nó và dừng/xóa nó khi cần thiết.

III. Chuẩn đầu ra

- Sinh viên nắm vững và sử dụng thành thạo class Process của .NET để quản lý process.
- Sinh viên hiểu rõ 1 trong 3 cách kích hoạt/dừng process được trình bày trong silde bài giảng: process đang chạy có thể kích hoạt tự động process khác chạy và khi cần có thể giết chết process khác.

IV. Qui trình

1. Giới thiệu

Ở chế độ mono-programming, từng thời điểm chỉ có 1 process được nạp vào bộ nhớ và thực thi. Khi process hiện hành đã hoàn thành, HĐH mới phục vụ process khác nếu có.

Ở chế độ multi-programming, từng thời điểm có thể có nhiều process được nạp vào bộ nhớ và được chạy đồng thời (theo cảm nhận của người dùng). Tuy nhiên, về mặt vật lý, từng thời điểm chỉ có 1 process thực sự chiếm CPU và đang chạy.

Ta nói 1 process đang ở trạng thái Ready nếu nó không chờ thiết bị I/O. Các process Ready sẽ được chạy luân phiên nhau, khi cần thiết ta sẽ thực hiện chuyển ngữ cảnh để thay đổi process chạy. Có 2 cơ chế thực hiện Multi-programming: theo lô (batch) hay phân chia thời gian.

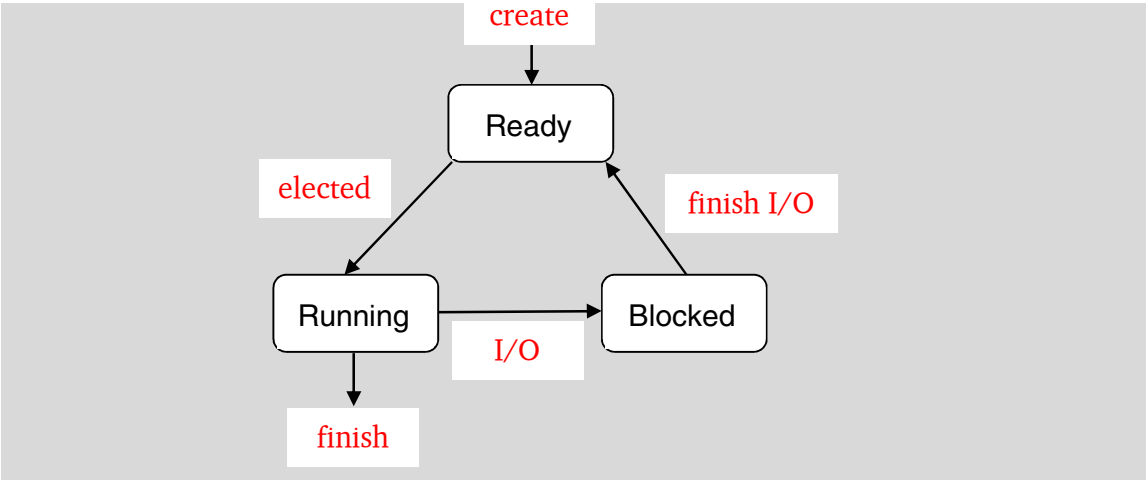
Trong cơ chế theo lô, việc chuyển ngữ cảnh chỉ xảy ra khi process hiện hành thực hiện hoạt động I/O và phải chờ I/O. Trong cơ chế phân chia thời gian, việc chuyển ngữ cảnh sẽ xảy ra khi 1 trong 2 trường hợp sau xảy ra: **1.** process hiện hành thực hiện hoạt động I/O và phải chờ I/O, **2.** process hiện hành đã chạy hết khoảng thời gian qui định (quantum). Khe thời gian (quantum) thường rất nhỏ (từ 10 đến 100ms).

Ta dùng thuật ngữ overhead để miêu tả khoảng thời gian chuyển ngữ cảnh giữa 2 process. Đây là khoản thời gian bị thiệt (vô ích theo góc nhìn của các process), nên càng chiếm tỉ lệ nhỏ so với quantum càng tốt.

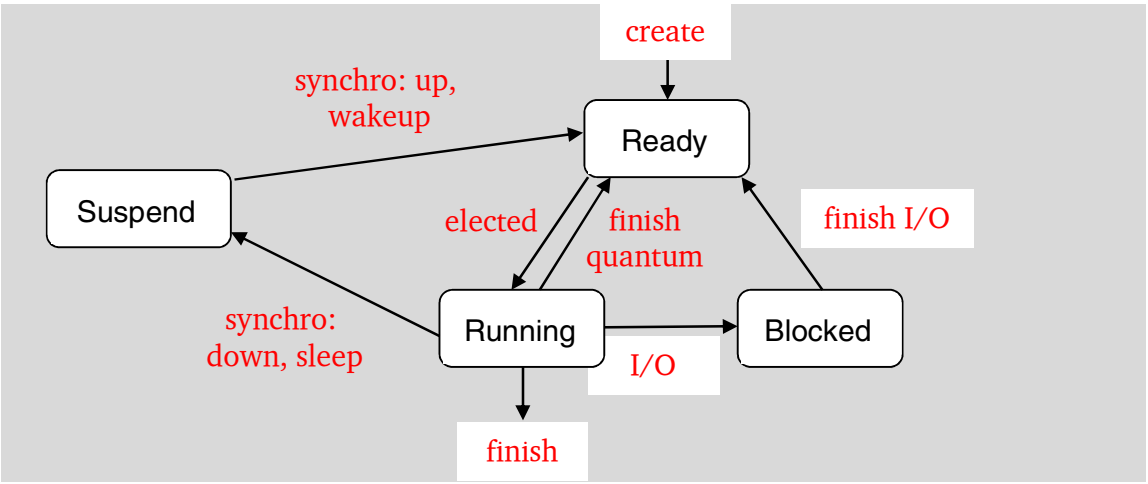
Bài thực hành này sẽ tìm hiểu ảnh hưởng của các chính sách quản lý CPU khác nhau. Giả sử mỗi máy tính chỉ có 1 CPU và 1 đơn vị I/O, bộ nhớ nội thì đủ lớn để chứa tất cả process cần chạy.

II. Sơ đồ chuyển trạng thái của từng process

Hãy vẽ sơ đồ chuyển trạng thái của từng process trong chế độ lô.



Hãy vẽ sơ đồ chuyển trạng thái của từng process trong chế độ phân chia thời gian, lưu ý mỗi process có thể bị mất CPU vì cần đồng bộ hóa với process khác.



III. Các tỉ lệ dùng CPU

Giả sử cần thi hành 3 tác vụ: T_1, T_2 và T3 với thông tin như sau:

T1 chạy tốn 200 ms (không kể thời gian I/O) và có thực hiện I/O 1 lần ở thời điểm 110ms sau khi chạy.

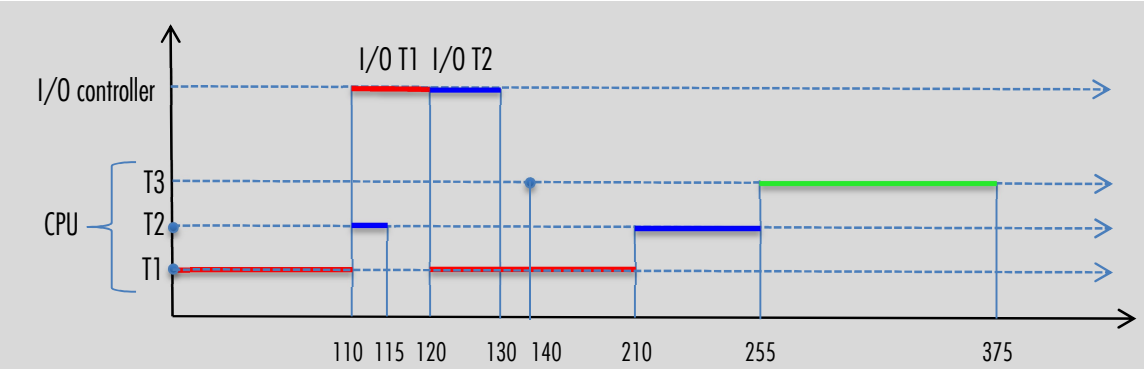
T2 chạy tốn 50 ms và có thực hiện I/O 1 lần ở thời điểm 5ms sau khi chạy.

T3 chạy tốn 120 ms và không thực hiện I/O lần nào.

T1 và T2 được kích hoạt tại thời điểm 0, T3 được kích hoạt tại thời điểm 140ms. Mỗi hoạt động I/O tốn 10 ms. Quantum là 100ms (dùng trong chế độ phân chia thời gian).

III.1 Vẽ lược đồ Gantt miêu tả việc thực thi 3 tác vụ

Tính tổng thời gian thực thi T và tỉ lệ chiếm giữ CPU τ_P ở chế độ lô:



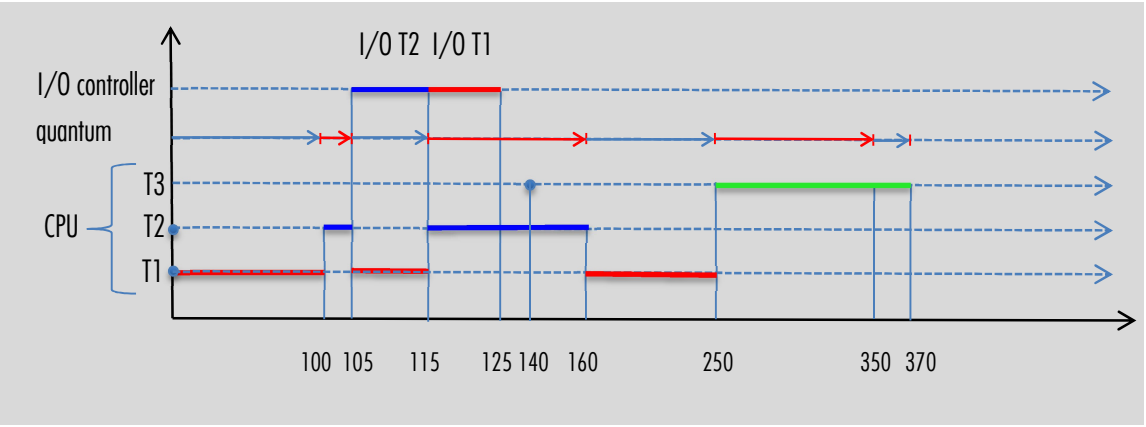
Ở t = 210, 2 process T2 và T3 đều Ready. T2 Ready từ lúc t = 130 còn T3 chỉ Ready từ lúc t =140, do đó ta cho T2 thi hành trước.

Tổng thời gian thực chạy cho các process T = 200+50+120 = 370ms

Tổng thời gian tốn thực tế là 375ms.

Tỉ lệ dùng CPU τ_P = 370/375 (tổng thời gian thực chạy cho các process/tổng thời gian tốn thực tế) = 98,7 %

Tính tổng thời gian thực thi T và tỉ lệ chiếm giữ CPU τ_P ở chế độ phân chia thời gian :



Tổng thời gian thực chạy cho các process T = 200+50+120 = 370ms

Tổng thời gian tốn thực tế là 370ms.

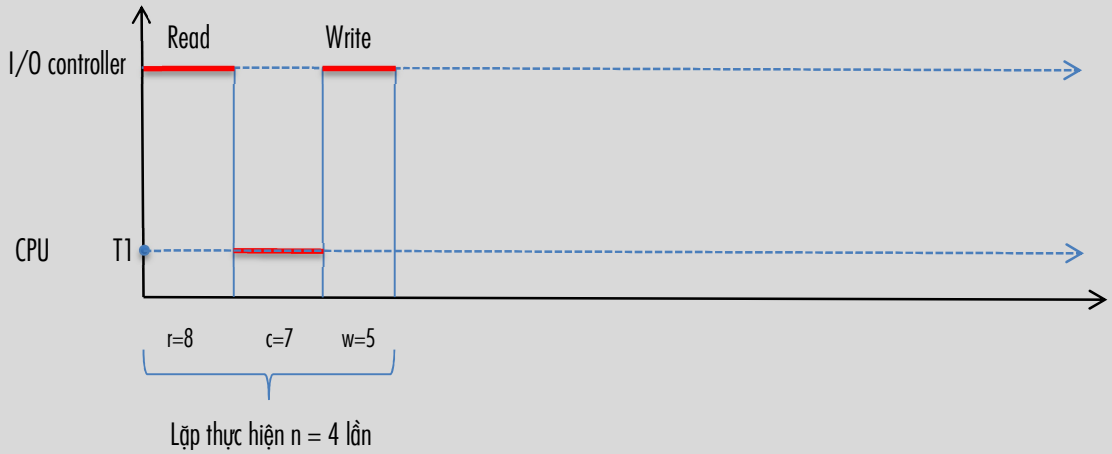
Tỉ lệ dùng CPU τ_P = 1

III.2 Vẽ lược đồ Gantt miêu tả việc thực thi 2 tác vụ

Hãy chú ý 2 tác vụ giống nhau T_1 và T_2, được kích hoạt đồng thời tại thời điểm t = 0 và thực hiện n lần công việc sau:

- đọc dữ liệu từ đĩa (tốn r ms)
- xử lý dữ liệu (tốn c ms)
- ghi kết quả lên đĩa (tốn w ms)

Nếu chỉ có T1 được kích hoạt chạy ở thời điểm t (T2 không có), hãy vẽ lược đồ Gantt miêu tả việc thực thi tác vụ T1, tính tổng thời gian thực chạy T của T1, tỉ lệ dùng CPU τ_P và dùng I/O τ_U. Ta có thể suy diễn T, τ_P et τ_U khi 2 tác vụ T1 và T2 cùng chạy ở chế độ monoprogramming? Cho w = 5 ms, c = 7 ms, r = 8 ms, n = 4.



Tổng thời gian tốn thực tế T: $T = n.(r + c + w)$

Tỉ lệ dùng CPU τ_P : $\tau_P = \frac{n.c}{T} = c / (r + c + w)$

Tỉ lệ dùng I/O τ_U : $\tau_U = (r + w) / (r + c + w)$

Thế các số liệu cụ thể vào các công thức, ta có :

Tổng thời gian tốn thực tế T : $T = 4.(8 + 7 + 5) = 80ms$

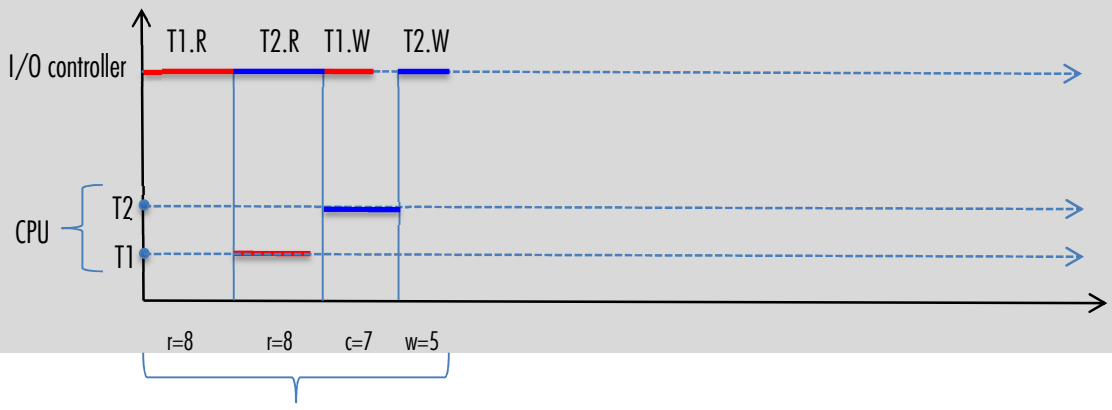
Tỉ lệ dùng CPU τ_P : $\tau_P = \frac{n.c}{T} = 7 / (8 + 7 + 5) = 8/20 = 35\%$

Tỉ lệ dùng I/O τ_U : $\tau_U = (r + w) / (r + c + w) = 13/20 = 65\%$

Nếu 2 tác vụ T_1 và T_2 được chạy ở chế độ monoprogramming, việc thi hành T2 sẽ đi ngay sau T1 và diễn biến giống hệt T1. Tổng thời gian tốn thực tế T sẽ nhân đôi. Tuy nhiên, τ_P và τ_U không đổi vì thời gian chạy CPU và truy xuất I/O cũng được nhân đôi.

Thế các giá trị được cho ta có: $T = 160$ ms $\tau_P = 35\%$ $\tau_U = 65\%$

Hãy vẽ lược đồ Gantt miêu tả việc thực thi 2 tác vụ T1 và T2 trong chế độ lô. Cho e = 5 ms, c = 7 ms, l = 8 ms, n = 4, như vậy w< c < r. Tính tổng thời gian thực chạy T, τ_P et τ_U cho 2 tác vụ T1 và T2. So sánh với chế độ monoprogramming.



Lặp thực hiện n = 4 lần

Lưu ý khi E1 kết thúc, mặc dù I/O rảnh trở lại nhưng T1 vẫn chưa thể thực hiện L1 cho chu kỳ kế tiếp được vì lúc này T1 đang dừng. Kết quả là T2 sẽ chiếm I/O trước để thực hiện hoạt động E2 như hình vẽ. Khảo sát hình vẽ ta xác định được :

$T = n. (2.r + c + w)$

$\tau_P = 2.c / (2.r + c + w)$

$\tau_U = 2.(r + w) / (2.r + c + w)$

Áp dụng các giá trị cụ thể ta có: $T = 112$ ms $\tau_P = 50\%$ $\tau_U = 93\%$

Nhờ sự chồng lấp 2 hoạt động dùng CPU và I/O (của 2 tác vụ), tỉ lệ dùng CPU và dùng I/O đã tăng lên, nghĩa là ta đã cải thiện độ hiệu quả việc dùng tài nguyên, kết quả là tổng thời gian thực chạy giảm xuống.