Like & Share & Subcribe

AnhNguyenNgoc ProIT4All

Giới thiệu

Process và Thread

Fan page: http://facebook.com/Proit4All

Lập trình đồng thời (concurrent programming)

- Trong lập trình đồng thời, có hai đơn vị thực thi cơ bản:
  - Process: tiến trình
  - Thread: luồng.
  - Java hỗ trợ Thread

Lập trình đồng thời (concurrent programming)

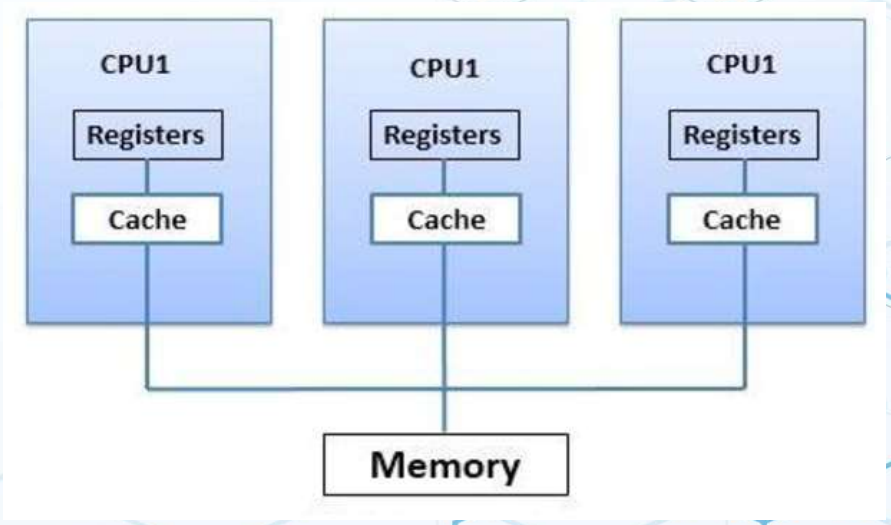
- Một hệ thống máy tính thường có nhiều Process và Thread đang hoạt động.
  - Điều này đúng ngay cả trong các hệ thống chỉ có một lõi thực thi.
  - Thời gian xử lý cho một lõi được chia sẻ giữa các Process và Thread qua một tính năng của hệ điều hành được gọi là Time Slicing (Lát cắt thời gian)
- Việc các hệ thống máy tính có nhiều bộ xử lý hoặc bộ xử lý có nhiều lõi thực thi ngày càng trở nên phổ biến.
  - nâng cao đáng kể khả năng của hệ thống để thực thi đồng thời các Process và Thread

Process là gì?

- Process là một tiến trình
  - Là môi trường thực thi khép kín và có thể xem như một thể hiện của chương trình hoặc ứng dụng đang được thực hiện.
  - Trong mỗi chương trình chứa nhiều Process bên trong.
  - JRE chạy dưới dạng một Process duy nhất chứa các lớp và chương trình khác nhau dưới dạng Process.

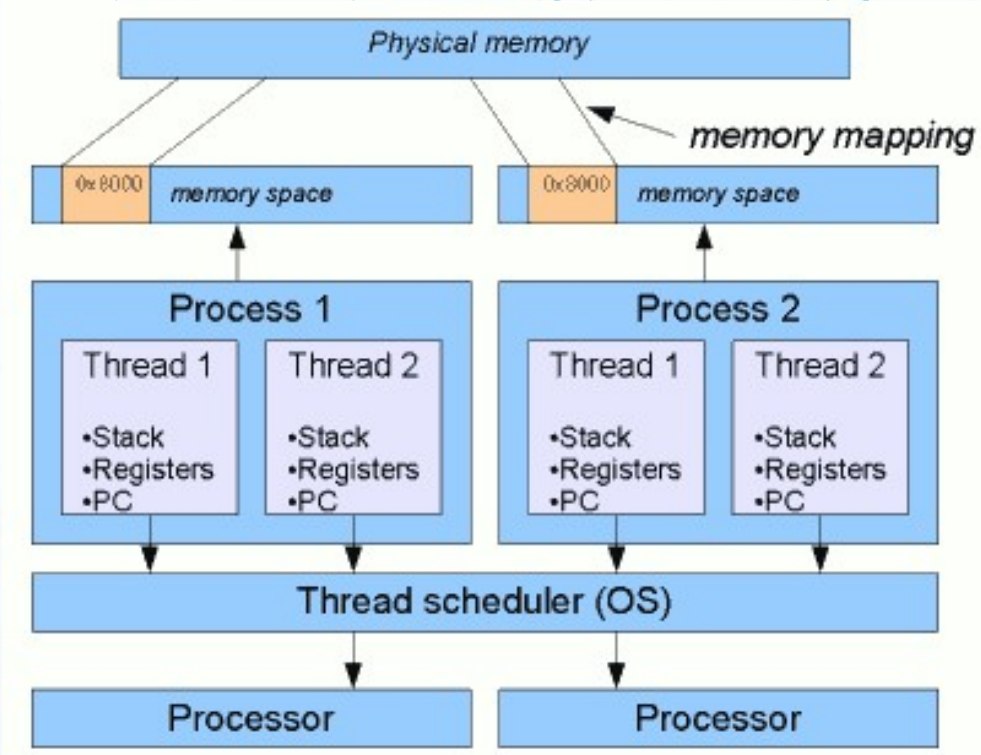
Multiprocess là gì?

- Multiprocessing là sự đa nhiệm dựa trên Process.
  - Nghĩa là khả năng nhiều chương trình hay ứng dụng có thể được thực hiện đồng thời.
- Ví dụ: Nghe nhạc, tải xuống tệp, Nhập notepad, v.v. từ internet cùng một lúc.
  - Tấc cả process đều được xử lý độc lập
  - Multiprocess được xử lý ở cấp OS (hệ điều hành).



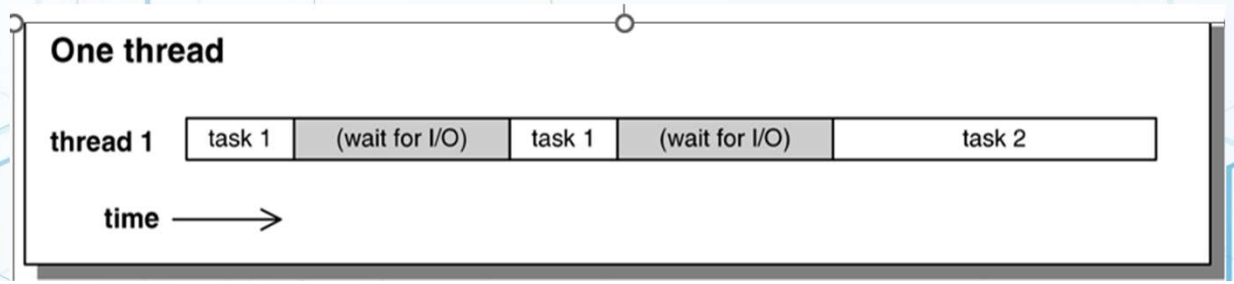
Thread là gì?

- Thread có thể được gọi là lightweight process (Tiến trình nhẹ).
  - Thread yêu cầu ít tài nguyên hơn để tạo và tồn tại trong Process
  - Thread chia sẻ tài nguyên của Process.

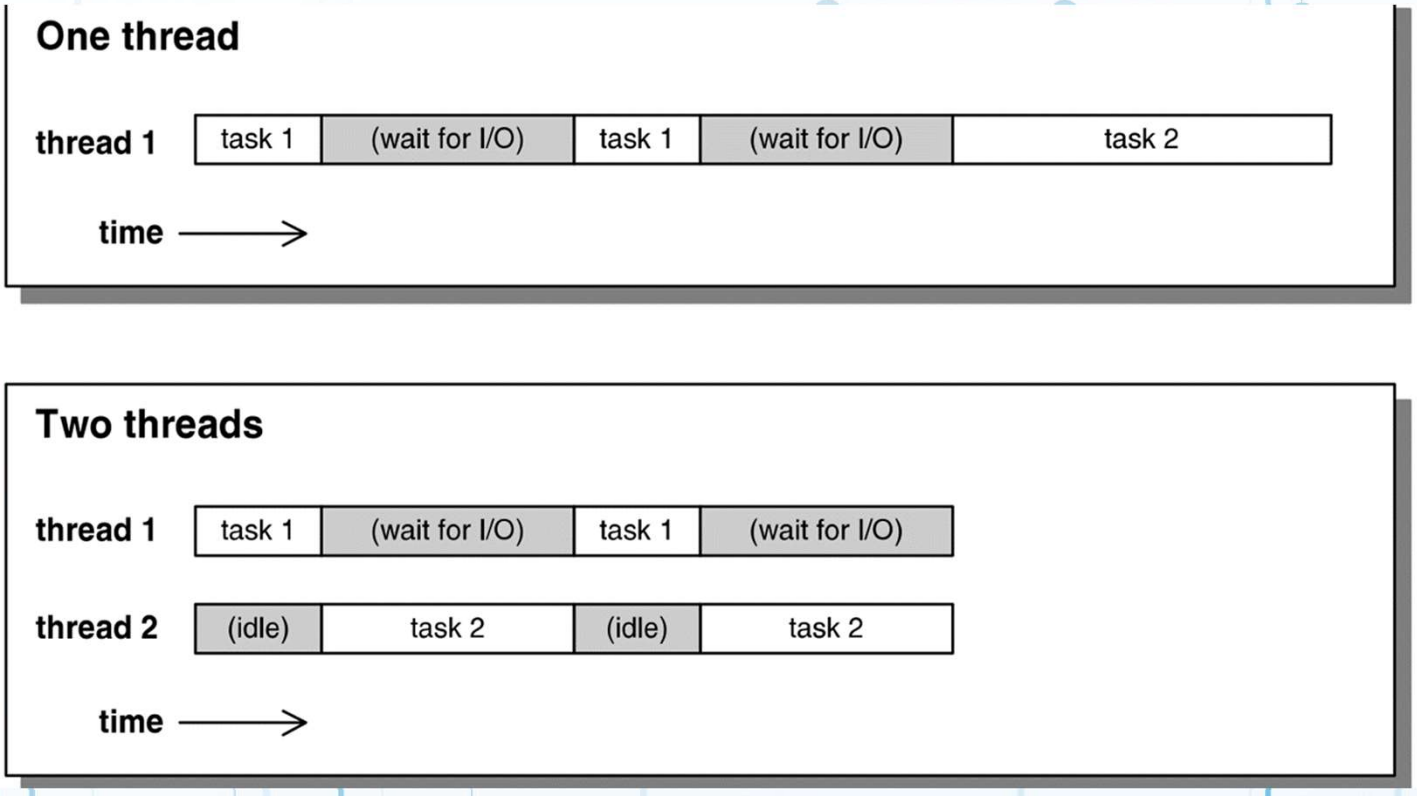


Thread trong Java

- Thread là một luồng thực hiện trong chương trình
- Mặc định, các ứng dụng Java sử dụng một luồng duy nhất, được gọi là main (luồng chính)
- Luồng điều khiển luôn tuần tự, mỗi lần một câu lệnh
- Trong một số trường hợp, thực thi đơn luồng có thể không hiệu quả



Multithreading là gì?



Multithreading là gì?

- Trong Java, Multithread (Đa luồng) đề cập đến một quá trình thực hiện đồng thời hai hoặc nhiều Thread để sử dụng tối đa CPU.
- Ưu điểm của đa luồng
  - Tiết kiệm thời gian vì có thể thực hiện nhiều thao tác cùng nhau.
  - Các Thread là độc lập, vì vậy không chặn người dùng thực hiện nhiều thao tác cùng một lúc và đồng thời, nếu một ngoại lệ xảy ra trong một Thread, sẽ không ảnh hưởng đến các Thread khác.

So sánh giữa Process và Thread

NO	Process	Thread
1.	Process là bất kỳ chương trình nào đang được thực thi.	Thread là một phân đoạn của một Process
2.	Process tốn nhiều thời gian để kết thúc	Thread tốn ít thời gian để kết thúc
3.	Tốn nhiều thời gian để tạo mới	Tốn ít thời gian để tạo
4.	mất nhiều thời gian hơn để chuyển ngữ cảnh.	mất ít thời gian hơn để chuyển ngữ cảnh.
5.	Process kém hiệu quả hơn về mặt giao tiếp.	Process hiệu quả hơn về mặt giao tiếp.
6.	Multiprogramming (Đa chương trình) hỗ trợ Multiprocess	Một process chứa nhiều Thread và hỗ trợ multithread (Đa luồng)
7.	Process bị cô lập nghĩa là mỗi process có không gian hoạt động độc lập so với process khác	Thread chia sẽ bộ nhớ trong cùng một Process
8.	Process được gọi là heavyweight process (Tiến trình nặng)	Thread là lightweight process (Tiến trình nhẹ) vì mỗi Thread trong Process chia sẻ mã, dữ liệu và tài nguyên

So sánh giữa Process và Thread

NO	Process	Thread
9.	Chuyển đổi Process sử dụng một giao diện trong hệ điều hành.	Chuyển đổi Thread không yêu cầu gọi hệ điều hành và tạo ra một ngắt (interrupt) đến kernel.
10.	Nếu một Process bị chặn thì sẽ không ảnh hưởng đến việc thực hiện các Process khác	Nếu một luồng mức người dùng bị chặn, thì tất cả các luồng mức người dùng khác cũng bị chặn.
11.	Quy trình có Process Control Block, Stack và Address Space riêng.	Chủ đề có PCB của cha mẹ, Thread Control Block, Stack và Address space chung.
12.	Các thay đổi đối với Process cha không ảnh hưởng đến các Process con.	Vì tất cả các Thread của cùng một Process chia sẻ không gian địa chỉ và các tài nguyên khác nên mọi thay đổi đối với Thread chính có thể ảnh hưởng đến hành vi của các Thread khác
13.	Process không chia sẽ dữ liệu với Process khác	Thread chia sẽ dữ liệu với Thread khác

Tại sao phải sử dụng Thread và MultiThread?

- Để cải thiện hiệu suất của ứng dụng với các thao tác I/O mở rộng
- Để cải thiện khả năng đáp ứng của các ứng dụng GUI
- Để cho phép hai hoặc nhiều người dùng chạy đồng thời các ứng dụng dựa trên máy chủ