Single and multithreaded processes:

* Phần lớn các phần mềm trong máy tính hiện đại đều có dạng multithreaded, tức đa luồng. Các ứng dụng trong máy tính đa phần đều chạy một process nhất định cùng với đó là nhiều luồng chạy bên trong.
* Các ứng dụng cũng có thể được thiết kế để tận dụng khả năng xử lý trên các hệ thống multicore, giúp thực hiện nhiều CPU task song song.
* Minh hoạ dễ hiểu cho tính ưu việt của sự đa luồng là trình xử lý Word có thể vừa in tài liệu sử dụng một thread nền, vừa cùng lúc chạy một thread khác nhận dữ liệu vào từ người dùng để gõ một văn bản mới.
* multithreaded cũng được sử dụng rộng rãi trong nhân hệ hiều hành (operating system kernels). Có nhiều luồng hoạt động trong một kernel, và mỗi luồng đảm nhiệm một công việc riêng biệt, như quản lý thiết bị, quản lý bộ nhớ, quản lý ngắt…

Slide 19: Một quy trình:

-Không gian địa chỉ (phần văn bản, phần dữ liệu)

-Chuỗi thực thi đơn

•Bộ đếm chương trình

•Các thanh ghi

•Stack

-Tài nguyên khác (các tệp mở, quy trình con)

Slide 20: Đa luồng là Nhiều luồng thực thi trong cùng một môi trường của quá trình

-Không gian địa chỉ (phần văn bản, phần dữ liệu)

-Việc thực thi của nhiều luồng, mỗi luồng có bộ riêng

•Bộ đếm chương trình

•Các thanh ghi

•Stack

-Tài nguyên khác (các tệp mở, quy trình con)

1. Slide 21: Sự khác nhau giữa luồng đơn và đa luồng: Một luồng đề cập đến việc thực thi toàn bộ quy trình từ đầu đến cuối mà không bị gián đoạn bởi một luồng trong khi đa luồng đề cập đến việc cho phép nhiều luồng trong một quy trình để chúng được thực thi độc lập trong khi chia sẻ tài nguyên của chúng.

2. Slide 22: bên phải là mục riêng tư, ứng với các luồng, bên trái là mục được chia sẻ bởi các luồng trong 1 quá trình

3. Slide 23: Lợi ích mà luồng mang lại

* 1. Lợi ích của Thread:
     + Multithread giúp các ứng dụng tương tác có thể hoạt động tốt hơn vì ngay cả khi một phần chương trình bị block hoặc cần một thời gian dài để hoạt động, chương trình nhìn chung vẫn có thể chạy. Bạn có thể hình dung: khi một chương trình đang chạy, và người dùng nhấn vào một nút lệnh mà cần rất nhiều thời gian để process, thì một hệ thống dạng single-threaded sẽ không kích hoạt bất kì hoạt động nào khác cho tới khi hoàn thành bước lệnh vừa rồi. Ngược lại, ứng dụng dạng multithread sẽ không làm gián đoạn quá nhiều quá trình của người dùng vì trong khi một thread được kích hoạt để thực hiện bước lệnh kia, một thread khác sẽ được kích hoạt để thực hiện bất kì bước lệnh ít tốn thời gian hơn mà người dùng yêu cầu
     + các tiến trình chỉ có thể chia sẻ dữ liệu thông qua các kĩ thuật như shared memory (vùng bộ nhớ chung) và message sharing (chia sẻ tin)
     + việc cung cấp tài nguyên và dữ liệu cho quá trình tạo process rất tốn kém. Như trong Solaris, tạo ra một process lâu hơn 30 lần so với tạo ra một thread trong process đó, và lâu hơn 5 lần so với tạo một context-switch.
     + multithread giúp các threads hoạt động song song trong các lõi xử lý khác nhau, trong khi đối với tiến trình dạng single-threaded, một thread chỉ có thể chạy trên một bộ xử lý, không quan trọng việc có bao nhiêu thread trong hệ thống hiện tại.

4. Slide 24,25,26: Các ví dụ về luồng:

Slide 24: 1 trình xử lí văn bản với 3 luồng: Nhiều bộ xử lý văn bản có tính năng tự động lưu toàn bộ tệp vào đĩa vài phút một lần để bảo vệ người dùng không bị mất công việc trong ngày trong trường hợp chương trình bị lỗi, hệ thống bị treo hoặc mất điện. Luồng đầu tiên chỉ tương tác với người dùng. Luồng thứ hai định dạng lại tài liệu khi được yêu cầu. Luồng thứ ba ghi nội dung của RAM vào đĩa theo định kỳ. (Many word processors have a feature of automatically saving the entire file to disk every few minutes to protect the user against losing a day's work in the event of a program crash, system crash, or power failure. The first thread just interacts with the user. The second thread reformats the document when told to. The third thread writes the contents of RAM to disk periodically.)

Slide 25: 1 máy chủ Web đa luồng: (This model allows the server to be written as a collection of sequential threads. The dispatcher's program consists of an infinite loop for getting a work request and handing it off to a worker. Each worker's code consists of an infinite loop consisting of accepting a request from the dispatcher and checking the Web cache to see if the page is present. If so, it is returned to the client and the worker blocks waiting for a new request. If not, it gets the page from the disk, returns it to the client, and blocks waiting for a new request.) Mô hình này cho phép máy chủ được viết như một tập hợp các luồng tuần tự. Chương trình của người điều phối bao gồm một vòng lặp vô hạn để nhận yêu cầu công việc và giao nó cho nhân viên. Mỗi mã của công nhân bao gồm một vòng lặp vô hạn bao gồm chấp nhận yêu cầu từ người điều phối và kiểm tra bộ nhớ cache của Web để xem trang đó có xuất hiện hay không. Nếu vậy, nó sẽ được trả lại cho máy khách và khối công nhân đang chờ một yêu cầu mới. Nếu không, nó sẽ lấy trang từ đĩa, trả lại cho máy khách và chặn để chờ một yêu cầu mới.

Slide 26: Phác thảo sơ bộ về mã cho slide trước :

1. Luồng gửi đi
2. Luồng làm việc

Vòng lặp chính của máy chủ Web nhận một yêu cầu, kiểm tra nó và thực hiện nó để hoàn thành trước khi nhận được yêu cầu tiếp theo. Trong khi chờ đĩa, máy chủ không hoạt động và không xử lý bất kỳ yêu cầu gửi đến nào khác. Nếu máy chủ Web đang chạy trên một máy chuyên dụng, như thường lệ, CPU chỉ đơn giản là không hoạt động trong khi máy chủ Web đang đợi đĩa. Kết quả thực là nhiều yêu cầu / giây ít hơn có thể được xử lý. Do đó, các luồng đạt được hiệu suất đáng kể, nhưng mỗi luồng được lập trình tuần tự, theo cách thông thường.(  The main loop of the Web server gets a request, examines it, and carries it out to completion before getting the next one. While waiting for the disk, the server is idle and does not process any other incoming requests. If the Web server is running on a dedicated machine, as is commonly the case, the CPU is simply idle while the Web server is waiting for the disk. The net result is that many fewer requests/sec can be processed. Thus threads gain considerable performance, but each thread is programmed sequentially, in the usual way.)

Slide 27:

Gói luồng mức người dùng: Loại luồng này được tải hoàn toàn trong không gian người dùng, kernel không biết gì về chúng. Khi luồng được quản lý trong không gian người dùng, mỗi tiến trình cần bảng luồng riêng của nó, bảng luồng bao gồm bộ đếm chương trình, con trỏ ngăn xếp, thanh ghi, trạng thái, v.v. Bảng luồng được quản lý bởi hệ thống thời gian chạy. Tạo một luồng mới, chuyển đổi giữa các luồng và đồng bộ hóa các luồng được thực hiện thông qua lệnh gọi thủ tục. tức là không có sự tham gia của nhân. Chủ đề Cấp độ người dùng nhanh hơn hàng trăm lần so với các chủ đề Cấp độ hạt nhân

Slide 28: Thư viện luồng, (hệ thống thời gian chạy) trong không gian người dùng

\*Luồng tạo

\*Luồng có sẵn

\*Luồng đệm

\*Vùng luồng (để tự nguyện từ bỏ CPU)  
Khối điều khiển luồng (TCB) (Bảng luồng) lưu trữ trạng thái của luồng người dùng (bộ đếm chương trình, thanh ghi, ngăn xếp)

* + - Kernel không biết sự hiện diện của luồng người dùng

Slide 29: Hệ điều hành truyền thống chỉ cung cấp một luồng nhân kernel mà được trình bày bởi PCB cho mỗi quy trình.

-Sự cố chặn: Nếu một luồng người dùng bị chặn -> luồng nhân bị chặn, -> tất cả các luồng khác trong quá trình đều bị chặn.

Slide 30: Trong luồng mức lõi, lõi thực hiện toàn bộ công việc chuyển động của luồng. Kernel có một bảng luồng theo dõi tất cả các luồng trong hệ thống. Khi một luồng muốn tạo một luồng mới hoặc phá hủy bất kỳ luồng nào hiện có, nó sẽ thực hiện một cuộc gọi Kernel, thực hiện hành động này.

Slide 31:

* Đa luồng được hệ điều hành hỗ trợ trực tiếp:

-Kernel quản lý các tiến trình và luồng

-Lập lịch CPU cho luồng được thực hiện trong kernel

Lợi thế của đa luồng trong kernel:

* Kernel có thể lên lịch đồng thời cho nhiều luồng từ cùng một quá trình trên nhiều quá trình.
* Nếu một luồng trong một tiến trình bị chặn, Kernel có thể lên lịch cho một luồng khác của cùng một tiến trình.
* Bản thân các quy trình hạt nhân có thể được đa luồng.

Nhược điểm của đa luồng trong kernel:

* Các luồng nhân thường chậm tạo và quản lý hơn các luồng người dùng.
* Việc chuyển quyền điều khiển từ luồng này sang luồng khác trong cùng một quy trình yêu cầu chuyển chế độ sang Kernel

Slide 32: Nhiều cách khác nhau đã được nghiên cứu để cố gắng kết hợp những ưu điểm của luồng cấp người dùng với luồng cấp nhân. Một cách là sử dụng các luồng cấp nhân và sau đó ghép các luồng cấp người dùng vào một số hoặc tất cả các luồng nhân

Trong thiết kế này, hạt nhân chỉ biết đến các luồng cấp hạt nhân và lên lịch cho các luồng đó. Một số luồng đó có thể có nhiều luồng cấp người dùng được ghép chồng lên trên chúng. Các luồng cấp người dùng này được tạo, hủy và lập lịch giống như các luồng cấp người dùng trong một quy trình chạy trên hệ điều hành mà không có khả năng đa luồng. Trong mô hình này, mỗi luồng cấp nhân có một số luồng cấp người dùng thay phiên nhau sử dụng nó.