HỆ ĐIỀU HÀNH
(OPERATING SYSTEM)

Trình bày:Nguyễn Hoàng Việt
Khoa Công Nghệ Thông Tin
Đại Học Cần Thơ

Chương 4: Luồng (Thread)

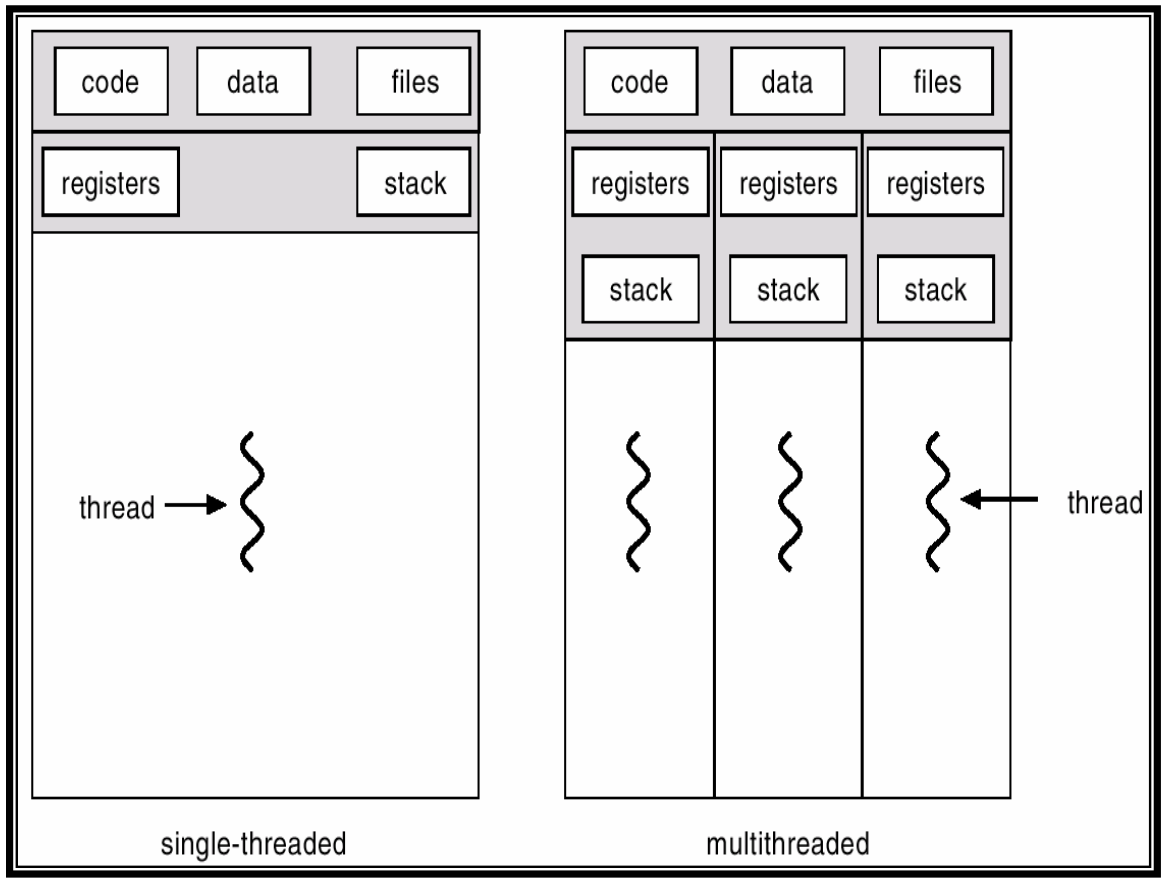
- Khái niệm cơ bản
- Các mô hình đa luồng (Multithreading Models)
- Các vấn đề phân luồng (Threading Issues)
- Pthreads
- Win32 threads
- Java threads

Các khái niệm cơ bản

- Quá trình có thể có:
  - Một luồng điều khiển hay một luồng hoạt động duy nhất
  - Nhiều luồng điều khiển hay nhiều luồng hoạt động
- Luồng là một dòng điều khiển bên trong một quá trình
  - Một đơn vị cơ sở cho việc sử dụng CPU
  - Bao gồm thread id, program counter, register set và stack
  - Nó chia sẻ với các luồng khác trong cùng quá trình mã lệnh, dữ liệu, và các tài nguyên khác của HĐH
- Các hệ thống đa luồng là phổ biến
  - Desktop PC
  - Web browser có thể có 2 luồng, một cho hiển thị và một cho góp nhặt thông tin

Các quá trình một và nhiều luồng

Các luồng trong cùng 1 quá trình chia sẻ với nhau mã lệnh, dữ liệu và tài nguyên khác của HĐH, ví dụ files.



Thuận lợi

- Gia tăng sự đáp ứng cho người dùng:
  - Một chương trình đang chạy với các luồng khác ngay cả nếu một phần của nó bi nghẽn (block) hoặc đang thực hiện một thao tác dài trong một luồng
- Chia sẻ tài nguyên
  - Các thread chia sẻ bộ nhớ và tài nguyên của quá trình
- Kinh tế
  - Tốn ít thời gian cho việc tạo và quản lý các luồng hơn các quá trình khi các luồng chia sẻ tài nguyên
  - Ví dụ, việc tạo luồng nhanh hơn 30 lần so với quá trình trong Solaris
- Sử dụng trong kiến trúc đa xử lý
  - Tăng mức độ song song bởi vì các luồng có thể chạy song song trên các bộ xử lý khác nhau

Các dạng luồng

- Hỗ trợ cho các luồng có thể được cung cấp ở mức người dùng (user level) hay mức nhân (kernel level)
- Luồng người dùng (User Thread):
  - Luồng được thực hiện bởi thư viện luồng mức người dùng
- Luồng mức nhân (Kernel Thread):
  - Được hỗ trợ và quản lý trực tiếp bởi HĐH ở mức nhân
  - Việc tạo, định thời và quản lý luồng được đặt trong không gian nhân
  - Chậm hơn trong việc tạo và quản lý
- Có mối quan hệ giữa luồng người dùng và luồng mức nhân

Thư viện luồng
(Thread Library)

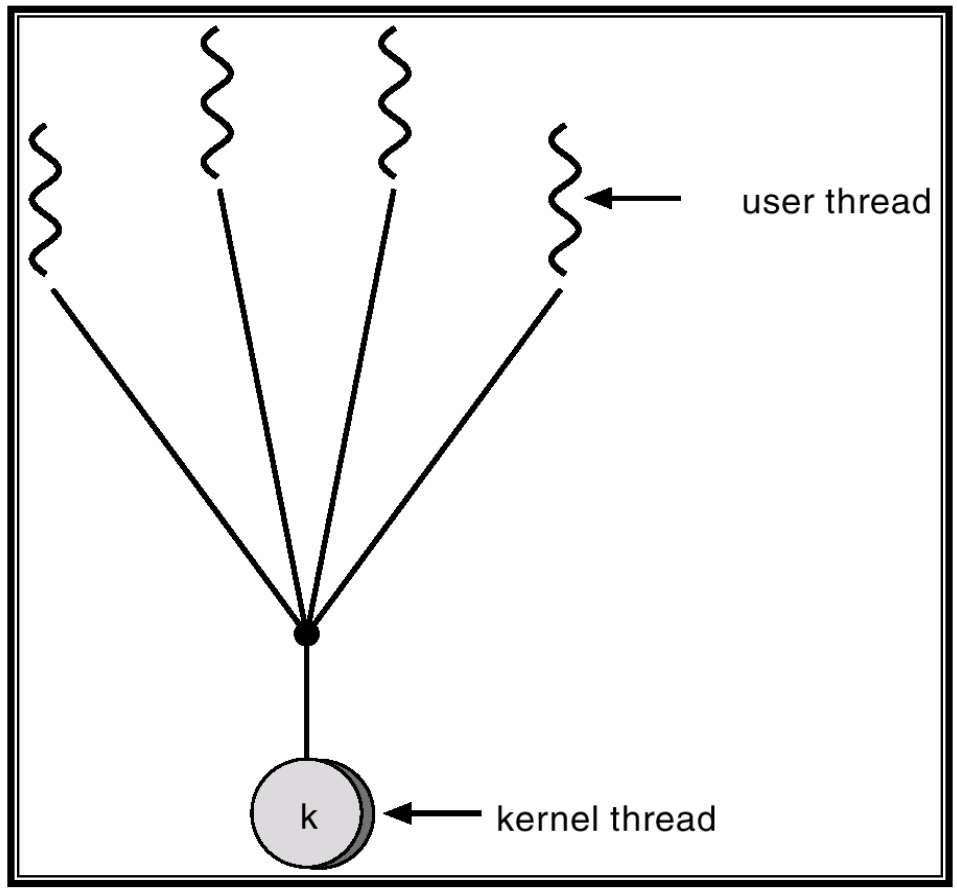
- Thread library cung cấp cho lập trình viên một API nhằm tạo và quản lý các luồng
- Hai phương pháp chính để cài đặt thread library (TL)
  - TL hoàn toàn trong không gian người dùng, không có hỗ trợ của nhân
    - Mã lệnh và cấu trúc dữ liệu của thư viện trong không gian người dùng
    - Thực hiện một function trong thư viện là một lời gọi hàm cục bộ trong không gian người dùng, không là lời gọi hệ thống
  - TL mức nhân, được hỗ trợ trực tiếp bởi HĐH
    - Mã lệnh và cấu trúc dữ liệu của thư viện trong không gian nhân
    - Thực hiện một function trong API cho thư viện là lời gọi hệ thống đến nhân
- Ba thread library hiện tại:
  - POSIX Pthreads: được cài đặt theo dạng 1 hay 2
  - Win32 thread library: thư viện mức nhân
  - Java thread API: có thể được cài đặt bởi Pthreads hoặc Win32 hoặc có thể bởi một thư viện khác

Các mô hình đa luồng (1)
(Multithreading Models)

- Ba phương pháp chung để thực hiện mối quan hệ giữa các luồng mức người dùng và luồng mức nhân
  - Nhiều-Một (Many-to-One)
  - Một-Một (One-to-One)
  - Nhiều-Nhiều (Many-to-Many)

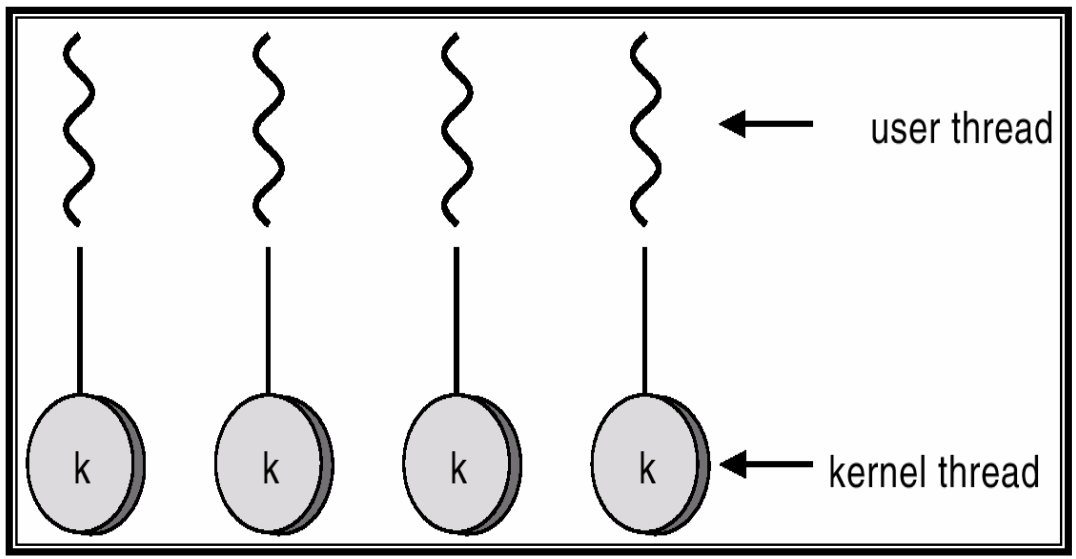
Các mô hình đa luồng (2)
Many-to-One

- Nhiều luồng mức người dùng ánh xạ vào một luồng mức nhân độc nhất
- Dễ dàng, hiệu quả trong việc quản lý luồng
- Vấn đề nghẽn
- Không đồng hành
- Ví dụ: các luồng màn hình (green threads) trong Solaris, , windows 95/97/98, OS/2



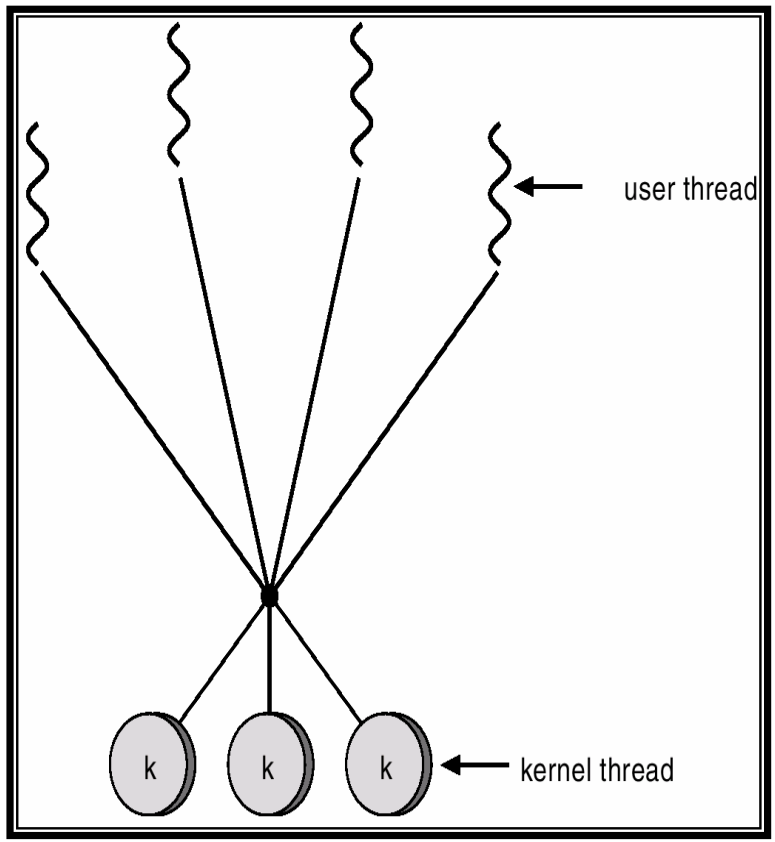
Các mô hình đa luồng (3)
One-to-One

- Mỗi luồng mức người dùng ánh xạ vào một luồng mức nhân
- Phí tổn cho việc tạo ra luồng mức nhân, một cho mỗi luồng người dùng
- Không bị nghẽn
- Cung cấp đồng hành
- Ví dụ: Linux, họ của Windows



Các mô hình đa luồng (4)
Many-to-Many

- Cho phép nhiều luồng người dùng ánh xạ vào nhiều luồng mức nhân
- Cho phép HĐH tạo ra đủ số luồng mức nhân
- Người dùng có thể tạo ra nhiều luồng người dùng như cần thiết và các luồng mức nhân tương ứng có thể chạy song song trên nhiều bộ xử lý
- Không nghẽn và cung cấp đồng hành
- Một biến thể là mô hình 2 mức: cho phép ánh xạ Many-to-Many và cả One-to-One
- Ví dụ: Solaris 2, Windows NT/2000 với ThreadFiber Package



Các vấn đề phân luồng (1)
(Threading Issues)

- Ngữ nghĩa của các lời gọi hệ thống fork() và exec()
- Hủy bỏ luồng (Thread Cancellation)
- Quản lý báo hiệu (Signal Handling)
- Các pool của luồng (Thread pools)
- Dữ liệu riêng của luồng (Thread-specific data)
- Các kích hoạt bộ định thời (Scheduler Activations)

Các vấn đề phân luồng (2)
Các lời gọi hệ thống fork() và exec()

- Thay về ngữ nghĩa của các lời gọi hệ thống fork() và exec()
- Hai ấn bản của lời gọi hệ thống fork()
  - Một ấn bản sao đúp chỉ luồng đã phát sinh ra lời gọi fork()
  - Một ấn bản khác sao đúp tất cả các luồng, ví dụ, sao đúp toàn bộ quá trình
- Lời gọi hệ thống exec không thay đổi
  - Chương trình được chỉ định trong tham số đến **exec()** sẽ thay thế toàn bộ quá trình – bao gồm tất cả các luồng
- Ấn bản nào của fork() được dùng tùy thuộc vào ứng dụng
  - Nếu exec() được gọi ngay sau fork(), việc sao đúp tất cả các luồng là không cần thiết, vì chương trình được chỉ định trong exec() sẽ thay thế quá trình ⇨ chỉ sao đúp luồng đã gọi
  - Quá trình riêng biệt không gọi exec() sau fork() sẽ sao đúp tất cả các luồng

Các vấn đề phân luồng (3)
Hủy bỏ luồng (Thread Cacellation)

- Tác vụ ngừng thực hiện một luồng trước khi nó hoàn thành
  - Hủy bỏ một luồng trong tìm kiếm đa luồng qua một cơ sở dữ liệu
  - Dừng một trang web đang load
- Luồng bị hủy bỏ thường được xem là luồng đích (target thread)
- Có 2 dạng hủy bỏ một luồng đích:
  - Hủy bỏ không đồng bộ (Asynchronous Cancellation)
    - Một luồng dừng ngay lập tức một luồng đích
  - Hủy bỏ trì hoãn (Deferred Cancellation)
    - Luồng đích có thể kiểm tra định kỳ để xem nó có thể dừng, cho phép luồng đích dừng cách có thứ tự theo cách của nó

Các vấn đề phân luồng (4)
Quản lý báo hiệu (Signal Handling)

- Báo hiệu cho một quá trình là một sự kiện nào đó đã xuất hiện
  - Bộ quản lý báo hiệu mặc nhiên hoặc do người dùng định nghĩa
- Báo hiệu đồng bộ (synchronous signal) liên hệ với một thao tác được thực hiện bởi một quá trình đang chạy
  - Truy cập bộ nhớ không hợp lệ hoặc phép chia cho 0
- Báo hiệu không đồng bộ (asynchronous signal) được tạo ra bởi một sự kiện bên ngoài quá trình đang chạy
  - Dừng một quá trình (Ctrl+C) hoặc quá hạn thời gian (timer expire)
- Các tùy chọn cho báo hiệu phát ra trong một quá trình đa luồng:
  - Báo hiệu cho luồng mà báo hiệu sẽ áp dụng với nó
  - Báo hiệu cho tất cả các luồng trong quá trình
  - Báo hiệu cho một số luồng xác định trong quá trình
  - Gán một luồng xác định để nhận tất cả báo hiệu cho quá trình

Các vấn đề phân luồng (5)
(Thread Pools)

- Tạo một số luồng lúc khởi động quá trình và đặt chúng vào một pool, nơi chúng chờ để làm việc
  - Ví dụ, đa luồng cho web server
- Một luồng trong pool được kích hoạt theo yêu cầu, và nó trở lại pool khi hoàn thành tác vụ
- Khi một yêu cầu đến trong tình trạng không có luồng nào sẵn dùng trong pool, server sẽ chờ cho đến khi một luồng tự do
- Thuận lợi của thread pool:
  - Phục vụ nhanh hơn vì không cần tạo ra luồng mới
  - Giới hạn về số luồng, tùy theo nhu cầu và tài nguyên hệ thống
- Kiến trúc thread-pool cho phép điều chỉnh tự động kích thước pool

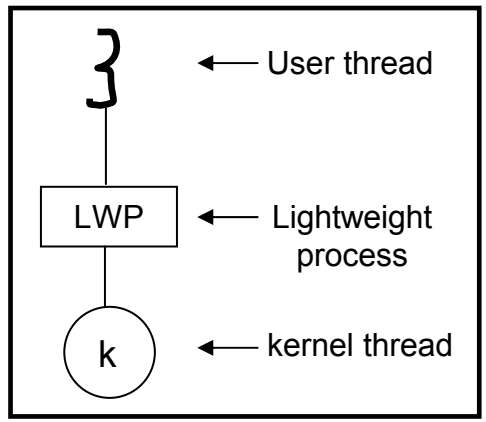
Các vấn đề phân luồng (6)
Dữ liệu riêng của luồng (Thread-Specific Data)

- Thread-specific data: dữ liệu riêng biệt của một luồng, không chia sẻ với các luồng khác
  - Ví dụ, đa luồng cho web server
- Ví dụ về thread-specific data
  - Trong một hệ thống xử lý giao dịch, các dịch vụ giao dịch khác nhau sẽ được cung cấp bởi các luồng khác nhau
  - Mỗi giao dịch sẽ được gán một nhận dạng duy nhất (unique identifier)
  - Có thể dùng thread-specific data để thực hiện gán này

Các vấn đề phân luồng (7)

Các kích hoạt bộ định thời (Scheduler Activations)

- Nhiều hệ thống cài đặt một cấu trúc dữ liệu trung gian, gọi là LWP (lightweight process)
  - Đối với user-thread library, LWP xuất hiện như bộ xử lý ảo mà các ứng dụng có thể định thời một luồng người dùng chạy trên nó
  - Mỗi LWP nối kết với một luồng mức nhân
  - Luồng mức nhân tương ứng chạy trên bộ xử lý thực



- Scheduler Activation: giao tiếp giữa thư viện luồng người dùng và nhân
  - Mỗi ứng dụng nhận một tập hợp các bộ xử lý ảo (LWPs) từ nhân
  - Ứng dụng định thời các luồng trên các LWP này
  - Nhân phải thông báo cho một ứng dụng về sự kiện nào đó. Thủ tục này được gọi là upcall và được quản lý bằng upcall handler trong thread library
  - Kernel sẽ tạo một upcall đến ứng dụng để thông báo có một luồng sắp nghẽn và nhận dạng luồng này. Sau đó nó cấp cho ứng dụng một LWP mới
  - Ứng dụng sẽ chạy một upcall handler trên LWP mới, lưu lại trạng thái của luồng nghẽn, ngưng sử dụng LWP được dùng bởi luồng nghẽn, cấp LWP mới này cho một luồng khác được chọn

Pthreads (1)

- Chuẩn của POSIX (IEEE 1003.1c) định nghĩa API cho việc tạo và đồng bộ hóa luồng
- API xác định cách xử lý của thư viện luồng
- Một tập hợp các dạng lập trình ngôn ngữ C và các lời gọi thủ tục
- Được cài đặt với tập tin header/include `pthread.h` và một thư viện luồng (thread library)
- Phổ biến trong HĐH UNIX

Pthreads (2)

Một ví dụ cài đặt đa luồng với Pthreads API

Hai luồng: luồng khởi tạo trong hàm **main** và một luồng mới thực hiện phép tổng trong hàm **runner**

```
# include <pthread.h>
void *runner(void *param);
main (int argc, char *argv[1]){
    pthread_t tid;
    pthread_attr_t attr;

    pthread_attr_init(&attr);
    pthread_create(&tid, &attr, runner, argv[1]);

    pthread_join(tid, Null);
}
```

```
void *runner(void *param)
{
  int upper = atoi(param);
  int I;
  sum = 0;
  if (upper > 0){
    for (I = 1; I <= upper; I++)
      sum += i);
  }
  pthread_exit(0);
}
```

Java Threads (1)

- Tạo ra một luồng mới được bắt nguồn từ Thread class

  `Thread runner = new Worker1();`
- Ghi đè lên một method đang chạy của một Thread class (trường hợp này là **run()**), bằng cách gọi method start()

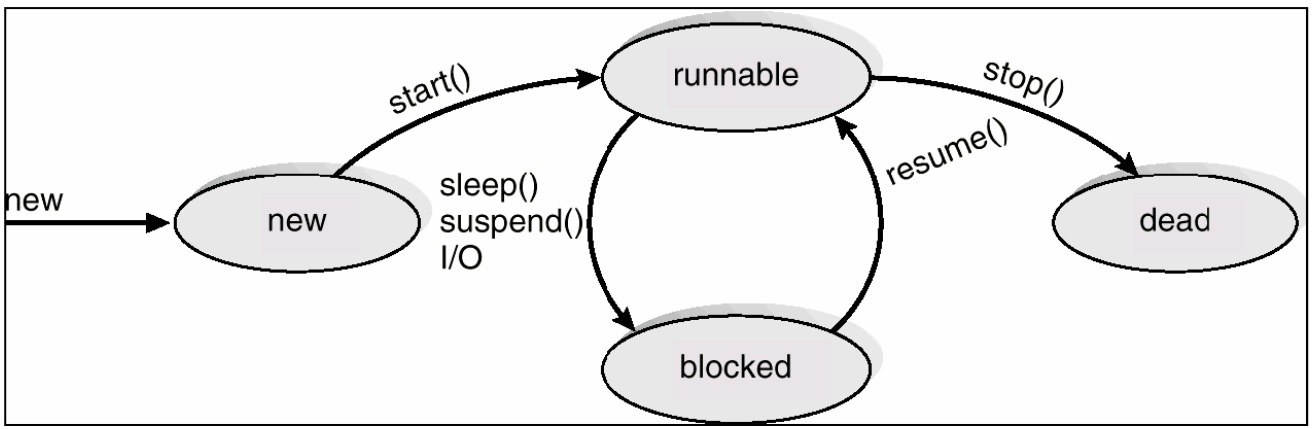
  `runner.start();`
- Khi chương trình chạy, hai luồng được tạo ra bởi JVM:
  - Luồng gắn kết với ứng dụng, khởi động thực thi tại method **main()**
  - Luồng runner, được tạo ra một cách tường minh với method **start()**, và bắt đầu thực hiện trong method **run()** của nó

```
class Worker1 extends Thread
{
    public void run() {
        System.out.println("I am a worker thread");
    }
}
public class First
{
    public static void main(String args[]) {
        Thread runner = new Worker1();
        runner.start();
        System.out.println("I am the main thread");
    }
}
```

Thread creation by extending the Thread class

Java Threads (2)

- Bốn trạng thái của luồng:
  - New: luồng ở trạng thái này khi object cho luồng được tạo ra
  - Runnable: gọii method start() cấp phát bộ nhớ cho luồng mới trong JVM và gọi method run() cho object của luồng. Khi đó nó sẵn sàng để chạy trong JVM
  - Blocked: luồng ở trạng thái này khi thực hiện một phát biểu nghẽn, ví dụ thực hiện thao tác I/O hoặc thực hiện một Thread method như sleep()
  - Dead: luồng chuyển sang trạng thái này khi method run() chấm dứt



Win32 Threads

- Windows XP cài đặt Win32 API và các ánh xạ One-to-One và Many-to-One (kèm với fiber library)
- Cấu trúc dữ liệu cho một thread:
  - ETHREAD (executive thread lock)
    - Con trỏ đến quá trình của nó
    - Con trỏ đến KTHREAD tương ứng
    - Địa chỉ của routine mà luồng đã khởi động trong đó
  - KTHREAD (kernel thread block)
    - Thông tin về định thời và đồng bộ hóa
    - Kernel stack
    - Con trỏ đến TEB
  - TEB (thread environment block)
    - Cấu trúc dữ liệu không gian người dùng cho luồng mức người dùng
    - User stack
    - Array of thread-specific data (Windows XP gọi là thread-local storage)

Linux Threads

- Lời gọi hệ thống fork() dùng sao đúp một quá trình, và clone() dùng tạo ra một luồng
- clone() xử lý giống như fork(),
  - nhưng thay vì tạo ra một bản sao của quá trình gọi, nó tạo ra một quá trình riêng biệt
  - dùng chung không gian địa chỉ của quá trình gọi ⇨ cho phép xử lý giống như một luồng riêng biệt
  - khi dùng clone(), một quá trình mới được tạo ra, nhưng thay vì sao chép cấu trúc dữ liệu của quá trình cha, quá trình mới này sẽ chỉ đến cấu trúc dữ liệu của quá trình cha
- Thực tế, Linux không phân biệt giữa quá trình và luồng, nó chỉ dùng thuật ngữ tác vụ (task).