**Mục Lục**

**1.Tiến trình**

-Tiến trình là gì

-Khởi tạo tiến trình

-Mô hình chuyển trạng thái tiến trình

**2.Luồng**

-Luồng là gì

-Cấu trúc dữ liệu của 1 luồng

**1. Tiến Trình**  
a. Khái niệm tiến trình  
- Một tiến trình bao gồm một tập các tài nguyên sử dụng khi thực thi một chương trình.  
Một tiến trình thường bao gồm các thành phần sau:  
- Một không gian địa chỉ ảo dành riêng, gồm những địa chỉ ảo mà tiến trình có thể sử  
dụng  
- Một chương trình thực thi, trong đó có mã, dữ liệu và được ánh xạ vào không gian địa  
chỉ ảo của tiến trình.  
- Một danh sách các handle của các tài nguyên, bao gồm semaphore, các cổng, các tệp tin  
- Một ngữ cảnh bảo mật được gọi là access token, định nghĩa quyền hạn của người dùng  
hay nhóm người dùng được liên kết với tiến trình, sẽ được nói đến trong phần 2.3.  
- Một số duy nhất để xác định tính duy nhất của tiến trình: process ID  
- Một hoặc nhiều luồng thực thi.  
- Mỗi tiến trình trỏ vào tiến trình cha của nó, nếu như không có tiến trình cha thì cũng  
không quan trọng vì Windows không quan tâm đến thông tin này và nó không ảnh hưởng  
đến hoạt động của hệ thống. Các thông tin về tiến trình có thể xem bởi công cụ Process  
Explorer của Sysinternal.com  
b. Cấu trúc dữ liệu  
Mỗi tiến trình trong Windows được biểu diễn dưới dạng một khối tiến trình thực thi  
(EPROCESS). Mỗi khối EPROCESS trỏ đến một số các cấu trúc dữ liệu liên quan khác  
như khối các luồng (ETHREAD – Chi tiết ở mục 2.3). Khối EPROCESS tồn tại trong

không gian địa chỉ hệ thống, EPROCESS liên kết với khối Môi trường tiến trình (PEB)  
nằm trong không gian địa chỉ tiến trỡnh(Vỡ nó chứa các thông tin mà được thay đổi bởi

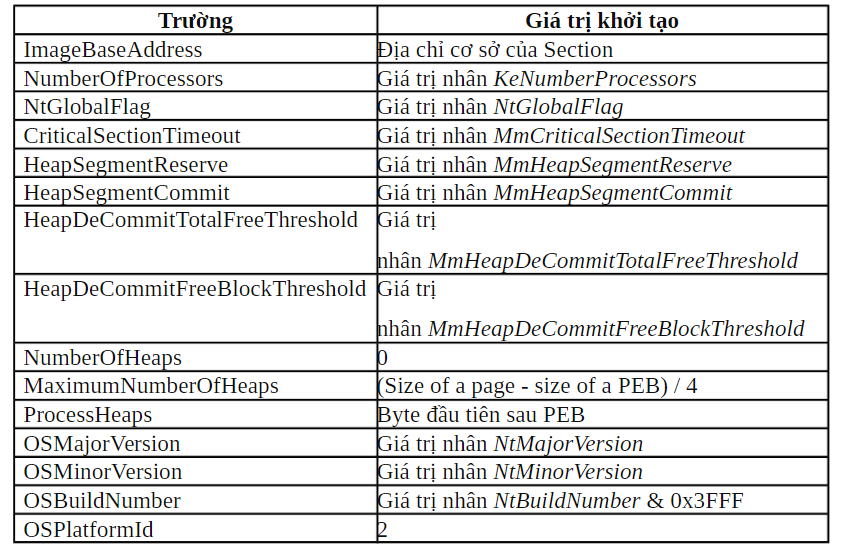
ứng dụng ở user-mode). Ngoài ra một khối EPROCESS còn trỏ đến Khối tiến trình của  
Windows và Bảng điều khiển handle.  
- Quá trình tạo một tiến trình  
Một tiến trình Windows được tạo khi mà ứng dụng gọi hàm tạo tiến trình, như là hàm  
Create Process, Create Process As User, Create Process With Token Who hoặc Create  
Process With LogonW. Để tạo một tiến trình thì cần những thông tin trong thư viện clientserver Kernel32.dll, trình thực thi của Windows và tiến trình hệ thống con của Windows.  
Các bước tạo một tiến trình mới:  
Để tạo một tiến trình với hàm API CreateProcess thì phải qua 6 bước cơ bản sau:  
B1: Mở tệp tin thực thi (.exe)  
B2: Tạo đối tượng thực thi tiến trình  
B3: Tạo luồng khởi tạo và stack, ngữ cảnh của nó.  
B4: Thông báo cho hệ thống con của Windows về tiến trình mới được tạo.  
B5: Bắt đầu thực thi luồng khởi tạo  
B6: Trong ngữ cảnh của luồng và tiến trình mới, hoàn thành việc khởi tạo của không gian  
địa chỉ(mục đích để nạp những thư viện liên kết động DLL) và bắt đầu thực thi chương  
trình.  
Trước khi gọi image, hàm CreateProcess thực hiện những bước sau:  
- Trong hàm CreateProcess,mỗi thứ tự ưu tiên cho các tiến trình mới là - một bít độc lập  
trong cờ CreationFlags, do đó có thể tạo một tiến trình - có nhiều mức ưu tiên, Windows  
sẽ xem xét và chọn thứ tự ưu tiên từ thấp đến cao để gán cho tiến trình mới tạo.  
- Nếu không có một thứ tự ưu tiên nào thì mặc định sẽ được đặt là Nornal.  
- Nếu ứng dụng có mức ưu tiên là Real-time và tiến trình gọi không có khả năng Nâng  
quyền ưu tiên, thì tiến trình mới tạo ra sẽ được gán mức ưu tiên là mức Cao.  
- Tất cả các tiến trình tạo ra đều được gắn với 1 desktop nào đó.

Bước 1: Mở tệp tin image  
Tệp image là tệp có khả năng chạy các tệp \*.exe, có nhiều loại tệp image như hình  
dưới đây, có nhiệm vụ tạo ra một đối tượng Section và ánh xạ nó vào không gian địa chỉ

bộ nhớ. Nếu không có tệp image nào được gọi thì mặc định sẽ gọi cmd.exe với tham số  
truyền sau đó là tên chương trình.  
Nếu ứng dụng trên Windows là tệp thực thi của Windows, thỡ nó sẽ được gọi trực tiếp  
luụn khụng thông qua chương trình image nào cả. Nếu tệp thực thi trong DOS như \*.com  
chẳng hạn thì Windows sẽ gọi tệp image Ntvdm.exe để chạy \*.com.  
Sau đó, nếu tệp thực thi là Windows exe thì CreateProcess sẽ đến bước 2, nếu là các  
tệp thực thi còn lại thì Bước 1 sẽ được khởi động lại, và quá trình thực hiện như sau:  
- Nếu tệp thực thi là MS-DOS với phần mở rộng là exe, com, pif, một thông điệp sẽ gửi  
đến cho hệ thống con Windows để kiểm tra xem đã chạy sẵn tệp image thực thi tương  
ứng chưa (Ntvdm.exe), các giá trị tham số được lưu trong  
HKLM\SYSTEM\CurrentControlSet\Control\WOW\cmdline. Nếu tệp image thực thi  
chưa được nạp thìCreateProcess sẽ quay lại bước 1. Nếu nạp rồi (Ntvdm.exe) thì sẽ  
chuyển qua bước 2.  
- Nếu tệp thực thi là MS-DOS có phần mở rộng là com hay bat thì tệp image thực thi  
tương ứng là Cmd.exe, tên của tệp thực thi đó sẽ được truyền dạng tham số cho Cmd.exe  
- Nếu tệp thực thi là Win16, CreateProcess sẽ quyết định VDM nào phải được tạo để nạp

tệp đó thông qua cờ điều khiển CREATE\_SEPARATE\_WOW\_VDM và  
CREATE\_SHARED\_WOW\_VDM. Nếu không có cờ nào được đặt thì mặc định sẽ gọi  
cờ HKLM\SYSTEM\CurrentControlSet\Control\WOW\ DefaultSeparateVDM. Sau khi  
VDM được tạo,CreateProcess sẽ tiếp tục nạp tệp thực thi đó. Nếu có một ứng dụng  
Win16 nữa được gọi, thì hệ thống con Windows sẽ gửi thông điệp xem VDM hiện tại có  
hỗ trợ không, nếu không thì CreateProcess sẽ chạy lại bước 1 để nạp tệp image thực thi  
tương ứng với các tham số như trên. Sau bước 1, CreateProcess đã mở được tệp image thực thi tương ứng với tệp cần chạy  
và tạo được một đối tượng Section cho nó. Đối tượng chưa được ánh xạ vào bộ nhớ,  
nhưng đã được mở. CreateProcess tìm trong HKLM\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Image File Execution Options để xem tên tệp thực thi đó cú ở đó  
chưa, nếu có ở đó thỡ nó sẽ chạy lại bước 1 với những tham số Debugger ở trong registry.  
Bước 2: Tạo Đối tượng tiến trình thực thi trong Windows.  
Để Tạo Đối tượng tiến trình thực thi trong Windows cần lời gọi hàm hệ  
thống NtCreateProcess, sẽ thực hiện các công việc con sau:  
2A: Khởi tạo khối EPROCESS  
2B: Khởi tạo không gian địa chỉ  
2C: Khởi tạo khối tiến trình của nhân KPROCESS  
2D: Ánh xạ tệp image thực thi vào không gian địa chỉ  
2E: Khởi tạo PEB  
2F: Hoàn thiện việc khởi tạo đối tượng tiến trình thực thi.  
Bước 2A: Khởi tạo khối EPROCESS  
- Cấp phát Windows EPROCESS  
- Kế thừa các thuộc tính từ tiến trình cha  
- Đặt kích thước tập các công việc  
vào PsMinimumWorkingSet và PsMaximumWorkingSet  
- Kế thừa tên của các thiết bị (ổ đĩa, COM port,…)  
- Lưu thông tin định danh của tiến trình cha vào InheritedFromUniqueProcessId  
- Tạo access token để quản lý truy nhập.  
- Đặt trạng thái thoát của tiến trình là STATUS\_PENDING.  
Bước 2B: Khởi tạo không gian địa chỉ  
- Tạo ra các trang trong những bản trang nhớ thích hợp để ánh xạ vào, số trang được tạolưu ở biến trong kernel MmTotalCommittedPages và nó sẽ đượccộngvàoMmProcessCommit.

Giá trị MmResidentAvailablePages sẽ được trừ đi tập các công việc nhỏ  
nhất(PsMinimumWorkingSet) để tính ra các trang nhớ đang còn trống.  
Bước 2C: Khởi tạo khối tiến trình của nhân KPROCESS  
Khởi tạo KPROCESS chứa những con trỏ đến một danh sách các luồng của hệ thống.  
KPROCESS cũng được trỏ đến thư mục các bảng trang nhớ(dựng để theo dõi không gian  
địa chỉ ảo của tiến trình), tổng thời gian mà các luồng đã được thực thi, thứ tự lên lịch  
chạy theo mức ưu tiên của tiến trình, CPU mặc định để thực thi các luồng trong tiến trình.  
Bước 2D: Ánh xạ tệp image thực thi vào không gian địa chỉ  
  
  
  
- Trình quản lý bộ nhớ ảo đặt giá trị của thời gian sẵn sàng của tiến trình thành thời gian  
hiện tại.  
- Trình quản lý bộ nhớ khởi tạo giá trị danh sách các công việc.  
- Ánh xạ đối tượng Section được tạo ở bước 1 vào không gian địa chỉ bộ nhớ mới. Địa  
chỉ cơ sở của tiến trình sẽ được đặt thành địa chỉ cơ sở của image.  
- Ntdll.dll được ánh xạ vào bộ nhớ  
Bước 2E: Khởi tạo PEB  
CreateProcess cấp phát trang nhớ cho PEB sau đó khởi tạo một số trường trong bảng:  
Bảng 2.5: Khởi tạo các trường trong PEB



Bước 2F: Hoàn thiện việc khởi tạo đối tượng tiến trình thực thi:  
- Nếu hệ thống cú cỏc thiết đặt về bảo mật thì quá trình tạo tiến trình sẽ được ghi vào tệp  
tin Security event log.  
- Nếu tiến trình cha có đối tượng công việc thì tiến trình con sẽ thêm đối tượng công việc  
này vào.

- Nếu như header của tệp image có đặt cờ IMAGE\_FILE\_UP\_SYSTEM\_ ONLY thì tất  
cỏc cỏc luồng trong tiến trình đó được chạy với 1 bộ xử lý duy nhất. Nếu không thì mỗi  
lần thực thi một luồng, bộ xử lý nào đang sẵn sàng thỡ nó sẽ được dùng (đối với hệ thống  
có nhiều bộ xử lý)  
- CreateProcess chèn khối tiến trình mới vào cuối của danh sách các tiến trình đang chạy  
trong Windows (PsActiveProcessHead);  
- Thời điểm mà tiến trình tạo ra được đặt lại, handle của tiến trình mới được chuyển cho  
Kernel32.dll  
Bước 3: Tạo luồng khởi tạo và stack, ngữ cảnh của nó  
Sau khi thực hiện xong bước 2, đối tượng thực thi đã được tạo ra, tuy nhiên chưa có  
luồng nào được tạo cả. Vì trước khi tạo luồng cần khởi tạo stack và ngữ cảnh để luồng có  
thể chạy được. Kích thước của stack là cố định bằng với kích thước trong tệp image.  
Lúc này, luồng sẽ được tạo ra bởi việc gọi hàm NtCreateThread. Các tham số trong  
luồng được lấy ra từ không gian địa chỉ của  
PEB. NtCreateThread gọi PspCreateThread để thực hiện các bước con sau:  
Tăng giá trị đếm số luồng trong đối tượng tiến trình lên 1  
Khởi tạo khối luồng thực thi ETHREAD  
Định danh của luồng được tạo ra cho luồng mới  
TEB khởi tạo không gian địa chỉ cho tiến trình ở User mode  
Địa chỉ bắt đầu của luồng ở user mode được lưu trong ETHREAD. Địa chỉ của luồng  
đầu tiên trùng vớiBaseProcessStart, còn các luồng tiếp theo thì địa chỉ bắt đầu  
từ BaseThreadStart.  
KeInitThread được gọi để thiết lập khối KTHREAD, thực hiện công việc như thiết đặt  
mức độ ưu tiên của luồng, cấp phát stack cho luồng, khởi tạo ngữ cảnh cho luồng. Sau đó  
KeInitThread gán trạng thái Initialied cho luồng và trả về cho PspCreateThread.  
Nếu có những thủ tục thông báo về việc tạo luồng thì sẽ được gọi Access token của luồng được thiết đặt giống như của tiến trình. Có thể dùng CreateRemoteThread để tạo luồng ở trong tiến trình khác, tuy nhiên phải xử lý access token xem tiến trình kia có cho phép tạo hay không.

Sau bước 3, luồng đã được khởi tạo và sẵn sàng để thực thi.  
Bước 4: Thông báo cho hệ thống con của Windows về tiến trình mới được tạo  
Kernel32.dll sẽ gửi thông điệp đến các hệ thống con Windows để cho các hệ thống  
này thiết đặt cho tiến trình mới và luồng mới. Thông điệp cú cỏc thông tin sau:  
- Handle của tiến trình và luồng  
- Các cờ tạo tiến trình  
- ID của trình tạo tiến trình  
Hệ thống con Windows sau khi nhận được thông điệp thì sẽ thực hiện các bước:  
- CreateProcess lặp lại handle của tiến trình và luồng lên 1  
- Khối tiến trình Csrss được cấp phát  
- Thiết đặt cổng cho tiến trình mới để hệ thống con Windows có thể nhận được các thông  
điệp xử lý ngoại lệ của tiến trình.  
- Khối luồng Csrss được cấp phát  
- CreateProcess chèn luồng vào danh sách luồng cho tiến trình.  
- Giá trị của số đếm các tiến trình tăng lên 1  
- Giá trị mặc định của Process Shutdown level được set thành 0x280  
- Khối tiến trình mới được chèn vào danh sách  
- Cấu trúc pre-process dùng bởi Windows kernel (W32PROCESS) được cấp phát và khởi  
tạo.  
- Ứng dụng khởi động con trỏ  
Bước 5: Bắt đầu thực thi luồng khởi tạo  
Luồng khởi tạo bắt đầu được thực thi nếu cờ CREATE\_SUSPENDED trong lúc tạo  
tiến trình không được thiết đặt.  
Bước 6: Thực thi tiến trình trong ngữ cảnh của tiến trình mới.

Một luồng bắt đầu được chạy ở kernel-mode bằng thủ tục KiThreadStartup, sau đó  
các tham số được truyền choPspUserThreadStartup để nạp image vào bộ nhớ bằng thủ  
tục LdrInitializeThunk trong Ntdll.dll. Thủ tục này hoàn thành nốt việc khởi tạo trình  
quản lý heap, bảng NLS(bảng hỗ trợ nhiều ngôn ngữ), mảng lưu trữ cục bộ của luồng và  
các thành phần quan trọng khác. Sau khi PspUserThreadStartup hoàn thành nó sẽ trả về

cho KiThreadStartup. APC dispatcher sẽ gọi hàm bắt đầu thực thi tiến trình năm ở user  
stack khi mà KiThreadStartup thực hiện xong.

-Trình bày mô hình chuyển trạng thái của tiến trình  
New: tiến trình đang được tạo lập.  
Running: các chỉ thị của tiến trình đang được xử lý

locked: tiến trình chờ được cấp phát một tài nguyên, hay chờ một sự kiện xảy ra .  
Ready: tiến trình chờ được cấp phát CPU để xử lý.  
Kết thúc: tiến trình hoàn tất xử lý.  
  
Tiến trình P1: vào hàng đợi Job-Queue ở trạng thái New, sẽ đợi 1 khoảng thời gian của  
quá trình điều phối chậm (Scheduler Long Term) của hệ điều hành(HĐH) để chọn tiến  
trình, sau khi được O.S chọn, P1 chuyển sang hàng đợi reday quueue và ở trạng thái  
Ready. Lúc này P1 chỉ đợi cấp CPU và running.  
Sau một khỏang thời gian running, tiến trình P2 xuất hiện. Lúc này, hệ điều hành sẽ ghi  
lại thông tin của P1 vào thanh PCB1 bao gồm những thông tin: con trỏ, trạng thái của  
P1, số hiệu của tiến trình P1, Bộ đếm P1, nội dung của P1…Và chuyển P1 sang hàng đợi  
Waiting và chuyển trạng thái Ready. Lúc này, P2 sẽ được cấp CPU và running. Và sau  
một khỏang thời gian running, P2 cũng sẽ chuyển sang hàng đợi waiting và chuyển trạng  
thái ready, lúc này HĐH cũng ghi lại thông tin vào thanh ghi PCB2 như đã làm ở P1.  
Sau đó, HĐH sẽ load lại thông tin của PCB1 và P1 sẽ tiếp tục running. Quá trình này  
cũng sẽ lập lại cho P2. Đển khi P1 và P2 kết thúc  
Tại một thời điểm, chỉ có một tiến trình có thể nhận trạng thái running trên một bộ xử  
lý bất kỳ. Trong khi đó, nhiều tiến trình có thể ở trạng thái blocked hay ready. Các cung

chuyển tiếp trong sơ đồ trạng thái biễu diễn sáu sự chuyển trạng thái có thể xảy ra trong  
các điều kiện sau :  
• Tiến trình mới tạo được đưa vào hệ thống  
• Bộ điều phối cấp phát cho tiến trình một khoảng thời gian sử dụng CPU  
• Tiến trình kết thúc  
• Tiến trình yêu cầu một tài nguyên nhưng chưa được đáp ứng vì tài nguyên chưa sẵn  
sàng để cấp phát tại thời điểm đó ; hoặc tiến trình phải chờ một sự kiện hay thao

tácnhập/xuất.  
• Bộ điều phối chọn một tiến trình khác để cho xử lý .  
• Tài nguyên mà tiến trình yêu cầu trở nên sẵn sàng để cấp phát ; hay sự kiện hoặc thao  
tác nhập/xuất tiến trình đang đợi hoàn tất.

- Phân tích vai trò của khối kiểm soát tiến trình

Khối kiểm soát tiến trình (Process Control Block - PCB ) - Bảng thông tin về môi  
trường và trạng thái hoạt động của tiến trình:  
  
Chứa các thông tin ứng với mỗi process. Process ID, parent process ID  
• Credentials (user ID, group ID, effective ID,...)  
• Trạng thái process : new, ready, running, waiting…  
• Program counter: địa chỉ của lệnh kế tiếp sẽ thực thi  
• Các thanh ghi CPU  
• Thông tin dùng để định thời CPU: priority,...  
• Thông tin bộ nhớ: base/limit register, page tables…  
• Thông tin thống kê: CPU time, time limits…  
• Thông tin trạng thái I/O: danh sách thiết bị I/O được cấp phát, danh sách các file đang  
mở,...  
• Con trỏ (pointer) đến PCBs khác

PCB đơn giản phục vụ như kho chứa cho bất cứ thông tin khác nhau từ quá trình này  
tới quá trình khác

**2. Khái niệm một luồng**  
Một luồng là một thực thể bên trong một tiến trình mà Windows lên lịch để thực thi,  
nếu không có luồng thì tiến trình không thể chạy được. Một luồng thường bao gồm:  
- Một tập các thanh ghi trạng thái của CPU  
- Hai stack, một dùng để cho luồng thực thi trên kernel mode và một dùng để thực thi trên  
user mode.  
- Một vùng nhớ riêng để lưu trữ dữ liệu, được gọi là TLS (thread-local storage) dùng để  
lưu trữ các thư viện  
- Định danh của luồng (thread ID)  
Các thanh ghi, stack, vùng nhớ riêng được gọi là ngữ cảnh của luồng (thread’s  
CONTEXT ). Những thông tin này thường khác nhau trên mỗi máy. Windows cung cấp  
hàm GetThreadContext để cung cấp thông tin cụ thể về ngữ cảnh này (CONTEXT  
block).  
Mặc dù các luồng có ngữ cảnh thực thi riêng, nhưng mỗi luồng trong cùng một tiến  
trình chia sẻ vùng không gian địa chỉ ảo của tiến trình đó, do vậy mà mỗi luồng có thể  
đọc/ghi bộ nhớ của luồng khác trong cùng một tiến trình. Các luồng không thể tham  
chiếu đến vùng không gian địa chỉ ảo của tiến trình khác, tuy nhiên, mỗi tiến trình có để  
ra một phần vùng địa chỉ riêng của nó làm vùng nhớ chia sẻ (được gọi là file mapping  
object trong hàm Windows API), hoặc một tiến trình có quyền để đọc ghi vào vùng nhớ  
của tiến trình khác sử dụng những hàm truy xuất bộ nhớ chéo như ReadProcessMemory

và WriteProcessMemory.  
a. Các luồng trong một đối tượng tiến trình  
Cả tiến trình và luồng đều có một ngữ cảnh bảo mật được lưu trong một đối tượng là  
access token. Mỗi access token của tiến trình đều chứa thông tin bảo mật cho tiến trình.  
Mặc định các luồng không có access token nhưng có thể 1 luồng trong số đó được gán  
một access token để bảo đảm an toàn cho nó

Bảng mô tả địa chỉ ảo (VAD) là một cấu trúc dữ liệu mà chương trình quản lý bộ nhớ  
sử dụng để theo dõi vùng không gian địa chỉ ảo mà tiến trình sử dụng.   
Cấu trúc dữ liệu của một luồng: Một luồng thường được biểu diễn bằng một khối  
luồng thực thi (ETHREAD). Khối này trỏ đến một không gian địa chỉ bộ nhớ hệ thống và  
khối môi trường luồng (TEB).  
b. Cấu trúc dữ liệu của một luồng

|  |  |
| --- | --- |
| Phần tử | Ý nghĩa |
| Thread time | Thời gian tạo và thoát luồng |
| Process ID | Định danh luồng |
| Start address | Địa chỉ bắt đầu |
| Impersonation | Trỏ đến access token để quản lý quyền hạn truy nhập |

Các hàm liên quan đến luồng

|  |  |
| --- | --- |
| CreateThread | Tạo một luồng mới |
| CreateRemoteThread | Tạo một luồng mới ở một tiến trình khác |
| OpenThread | Mở một luồng đã có |
| ExitThread | Kết thúc hoạt động của 1 luồng một cách bình thường |
| TerminateThread | Ngắt luồng |
| GetExitCodeThread | Lấy mã lúc thoát của một luồng khác |
| GetThreadTimes | Trả về thời gian chạy của một luồng |
| GetCurrentProcess | Trả về handle của luồng hiện tại |

Khi một luồng mới được tạo ra, nó cú một kernel stack riêng, trạng thái của luồng cũ  
sẽ được lưu vào đỉnh của stack của luồng cũ, và ngữ cảnh luồng sẽ nạp các thông tin của  
luồng mới vào kernel stack của nó. Nếu luồng nằm trong một tiến trình mới thì hệ thống  
sẽ tạo một trang nhớ mới và nạp địa chỉ của nó vào thanh ghi CR3. Địa chỉ trang nhớ có  
thể tìm thấy được trong khối KPROCESS. Nếu rootkit có thể thay đổi được bảng trang  
của tiến trình thỡ nó sẽ ảnh hưởng đến toàn bộ các luồng trong tiến trình đú, vỡ tất cả các  
luồng trong một tiến trình dùng chung 1 giá trị thanh ghi CR3

Trình bày mô hình luân chuyển CPU giữa hai tiến trình  
  
Phân biệt các loại trình điều phối  
Điều phối chậm (Long-term scheduler (or job scheduler)) :  
• Chọn process nào sẽ được đưa vào ready queue (từ New chuyển sang Ready)  
• Kiểm soát Độ đa chương  
• Do có nhiều thời gian (tới vài phút), loại scheduler này có điều kiện để lựa chọn kỹ càng  
nhằm phối hợp cân đối 2 loại tiến trình:  
Hướng CPU: tính toán nhiều, ít I/O

Hướng I/O: tính toán ít, nhiều I/O.  
• Mục đích cân bằng tải  
Điều phối nhanh (Short-term scheduler (or CPU scheduler)) :  
• Còn gọi là Điều phối CPU.  
• Chọn tiến trình từ Ready Queue để cấp CPU.  
• Có tần suất công việc cao. Thường cứ 100 ms lại tốn 10 ms để xác định tiến trình kế  
tiếp, như vậy 10/(100+10)=9% thời gian CPU được dùng để điều phối công việc.  
Điều phối vừa (Medium-term scheduler) :  
• Là Short-Term Scheduler được thêm chức năng rút các tiến trình khỏi bộ nhớ, dẫn đến  
làm giảm Độ đa chương, sau đó đưa lại chúng vào bộ nhớ vào thời điểm thích hợp để tiếp  
tục thực hiện từ vị trí bị tạm ngừng trước đó.  
• Nhờ cách điều phối này, hỗn hợp các tiến trình trong Ready Queue có tính tối ưu hơn.