TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

MÔN HỌC NGUYÊN LÝ HỆ ĐIỀU HÀNH

ĐỀ TÀI

NGHIÊN CỨU TÌM HIỂU VỀ QUẢN LÝ TIẾN TRÌNH TRONG HĐH WINDOWS

Giáo viên hướng dẫn: PhD. Nguyễn Bá Nghiễn

Sinh viên thực hiện: 1. Nguyễn Duy Linh

2. Nguyễn Đình Quang Huy

3. Nguyễn Quang Hưng

4. Phạm Quang Hưng

5. Trần Thị Thu Trang

Hà Nội, Ngày 14 tháng 6 năm 2022

MỤC LỤC

LỜI NÓI ĐẦU

Ngày nay cùng với sự phát triển vượt bậc của các ngành khoa học kĩ thuật thì ngành Công Nghệ Thông Tin (CNTT) đã và đang là ngành phát triển mạnh nhất. Nó đã đạt được nhiều thành tựu to lớn về khoa học và kĩ thuật để ứng dụng vào đời sống của con người ngày càng cao, cùng với sự cải tiến nâng cấp linh kiện thiết bị. Thì các chương trình phần mềm ứng dụng ra đời ngày nay càng tối ưu và ngày càng trợ giúp con người chúng ta giảm bớt gánh nặng công việc rất nhiều, sự phát triển phần mềm ứng dụng càng có tính chất quyết định đến sự phát triển của ngành khoa học CNTT đang còn non trẻ ở nước ta.

DANH MỤC HÌNH VẼ

CHƯƠNG 1: KHÁI NIỆM CHUNG VỀ HỆ ĐIỀU HÀNH WINDOWS.

1. Windows là gì ?

Là phần mềm hệ điều hành của hãng Microsoft.

Có giao diện đồ họa thông qua các hệ thống cửa sổ lệnh (Windows command)

2. Phần cứng(Hardware) là gì ?

Là các thành phần vật lý của máy tính, hay nói cách khác là các thành phần mà ta sờ vào được bằng tay.

VD : Bàn phím, chuột…

3. Phần mềm(Software) là gì ?

Là tập hợp những lệnh hướng dẫn máy tính hoạt động hay nói cách dễ hiểu là những gì chúng ta không sờ được bằng tay, VD: MS Word, Excell….

4. Chức năng cơ bản của Hệ điều hành là gì ?

Điều khiển tất cả hoạt động của máy tính và các thiết bị ngoại vi.

Đóng vai trò là người thông dịch, cầu nối giữa người sử dụng và máy vi tính thể thực hiện nhiều chức năng cùng 1 lúc thông qua các cửa sổ

Hệ điều hành Windows là hệ điều hành đa tác vụ, có nghĩa là nó có giao tiếp Windows.

Plug & Play, có nghĩa là tự động dò tìm và cài đặt các thiết bị gắn thêm vào hệ thống.

Ngoài ra nó còn cung cấp các tiện ích để kết nối mạng và Internet.

5. Ổ đĩa(drive) là gì ?

Là nơi để chứa chương trình và dữ liệu

6. Thư mục(Folder,Directory) là gì ?

Là phân vùng hình thức trên ổ đĩa để việc lưu trữ các tập tin được tổ chức

một cách có hệ thống.

7. Tập tin (file) là gì ?

Là một tập hợp các thông tin do người dùng tạo ra, các thông tin này là một hay nhiều chuỗi ký tự.

Quy tắc đặt tên cho tập tin (filename) là để phân biệt giữa các tập tin với nhau.

8. Đường dẫn (path) là gì?

Là đường chỉ đến một tập tin, hay một thư mục nào đó.

Đường dẫn tuyệt đối : là đường dẫn chỉ từ thư mục gốc đến tập tin

Đường dẫn tương đối : là đường dẫn của một đối tượng nhìn từ thư mục hiện hành.

CHƯƠNG 2: QUẢN LÝ TIẾN TRÌNH, LUỒNG VÀ CÔNG VIỆC

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 trõnh(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.

2. 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 W*ho hoặc *Create Process With LogonW*. Để tạo một tiến trình thì cần những thông tin trong thư viện client-server 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, thì 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 thì*CreateProcess* 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 thì 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 tạo lưu ở biến trong kernel *MmTotalCommittedPages* và nó sẽ được cộng vào *MmProcessCommit*.

- 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

Trường	Giá trị khởi tạo
ImageBaseAddress	Địa chỉ cơ sở của Section
NumberOfProcessors	Giá trị nhân *KeNumberProcessors*
NtGlobalFlag	Giá trị nhân *NtGlobalFlag*
CriticalSectionTimeout	Giá trị nhân *MmCriticalSectionTimeout*
HeapSegmentReserve	Giá trị nhân *MmHeapSegmentReserve*
HeapSegmentCommit	Giá trị nhân *MmHeapSegmentCommit*
HeapDeCommitTotalFreeThreshold	Giá trị nhân *MmHeapDeCommitTotalFreeThreshold*
HeapDeCommitFreeBlockThreshold	Giá trị nhân *MmHeapDeCommitFreeBlockThreshold*
NumberOfHeaps	0
MaximumNumberOfHeaps	(Size of a page - size of a PEB) / 4
ProcessHeaps	Byte đầu tiên sau PEB
OSMajorVersion	Giá trị nhân *NtMajorVersion*
OSMinorVersion	Giá trị nhân *NtMinorVersion*
OSBuildNumber	Giá trị nhân *NtBuildNumber* & 0x3FFF
OSPlatformId	2

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 thì 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 với*BaseProcessStart*, 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

cho*PspUserThreadStartup* 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.

3. 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 này được giải thích chi tiết ở phần 2.5.

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

Bảng 2.6 ý nghĩa các trường trong cấu trúc dữ liệu của 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 information	Trỏ đến access token để quản lý quyền hạn truy nhập
LPC information	Định danh của thông điệp cần được lấy địa chỉ
I/O information	Danh sách yêu cầu vào ra

Chi tiết về cấu trúc KTHREAD bên trong ETHREAD

Bảng 2.7 ý nghĩa các trường trong KTHREAD

Phần tử	**Ý nghĩa**
Dispatcher header	Header chuẩn của một khối dạng kernel dispatcher object.
Execution time	Thời gian dùng CPU cả ở user mode và kernel mode
Pointer to kernel stack information	Trỏ đến địa chỉ cơ sở của kernel stack
Pointer to system service table	Mỗi tiến trình bắt đầu đều phải trỏ đến bảng dịch vụ hệ thống *KeServiceDescriptorTable*
Scheduling information	Các thông tin lênh lịch chạy, bao gồm thứ tự ưu tiên, định lượng, các quan hệ, bộ xử lý, số lần treo, số lần dừng.
Wait blocks	Có sẵn một số khối đợi để khi luồng đợi 1 cái gì đó. 1 khối đợi ứng với 1 khoảng thời gian nào đó.
Wait information	Danh sách các đối tượng mà luồng cần phải đợi bao gồm đợi cái gì, bao lâu, lý do phải đợi
Mutant list	Danh sách các đối tượng mà luồng sở hữu
APC queues	Hàng đợi các APC ở user mode và kernel mode
Timer block	Bộ đếm giờ cho wait block
Queue list	Con trỏ đến đối tượng Hàng đợi mà luồng gắn với
Pointer to TEB	Chứa Định danh luồng, thông tin TLS, con trỏ PEB, GDI và OpenGL, chi tiết như hình dưới:

Một khối TEB chứa các thông tin về ngữ cảnh cho trình nạp image và các thư viện DLL của Windows khỏc. Vỡ cỏc thành phần đều chạy ở user mode nên cấu trúc dữ liệu này có thể ghi được trên user mode, do đó nó tồn tại trên

không gian địa chỉ của tiến trình thay vì tồn tại trong không gian địa chỉ của hệ thống. Muốn xem thông tin chi tiết về cấu trúc nào có thể dùng lệnh !thread của trình Kernel Debugge.

c. Cấu trúc khối TEB

Bảng 2.8: Các biến của Kernel quản lý việc tạo và thực thi luồng

Biến	Loại	Mô tả
PspCreateThreadNotifyRoutine	Mảng các con trỏ	Mảng các con trỏ đên những thủ tục sẽ được gọi khi tạo và xóa các luồng.
PspCreateThreadNotifyRoutineCount	DWORD	Số thủ tục thông báo về đăng ký luồng.
PspCreateProcessNotifyRoutine	Mảng các con trỏ	Mảng các con trỏ đến các thủ tục sẽ được gọi trong lúc tạo xóa tiến trình.

Bảng 2.9: Các hàm liên quan đến luồng

Hàm	Mô tả
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.
GetCurrentProcessId	Trả về định danh của luồng hiện tại
GetThreadId	Trả về định danh của 1 luồng bất kì
Get/SetThreadContext	Trả về thay đổi trong thanh ghi CPU của luồng.
GetThreadSelectorEntry	Trả về bảng mô tả luồng(chỉ có trong các hệ thống x86)

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 thì 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.

4. Kiểm tra hoạt động của một luồng

Để có thể xem thông tin của một luồng, sử dụng một tập các công cụ dưới đây:

Thuộc tính	Perfmon	Pviewer	Pstat	Qslice	Tlist	KD Thread	Process Explorer	Pslist
ThreadID	□	□	□		□	□	□	□
Actual start add	□	□	□			□	□	□
Win32 start add					□	□	□	
Current address	□	□				□	□	
Số context switches	□	□	□				□	□
Total user time		□	□			□	□	□
Total privileged time		□	□			□	□	□
Elapsed time	□	□				□	□	□
Thread state	□		□		□	□	□	□
Reason for wait state	□		□		□	□	□	□
Last error					□		□	
%CPU sử dụng	□			□			□	
%User time sử dụng	□			□			□	
%Previleged time sd	□			□			□	

Bảng 2.10 Các công cụ kiểm tra hoạt động của luồng.

5. Đối tượng Công việc

Một đối tượng công việc là một đối tượng của nhân cho phép điều khiển một nhóm có nhiều tiến trình. Một tiến trình chỉ có thể là thành viên của một đối tượng công việc duy nhất. Mặc định thì sự liên kết các tiến trình trong 1 đối tượng Công việc không thể phá hủy được, và tất cả các tiến trình được tạo bởi một tiến trình sẽ nằm trong đối tượng công việc mà tiến trình đó đang liên kết.

Bảng 2.11: Các hàm quản lý đối tượng Công việc

Hàm	**Mô tả**
CreateJobObject	Tạo một đối tượng công việc.
OpenJobObject	Mở đối tượng công việc có sẵn.
AssignProcessToJobObject	Thêm một tiến trình vào đối tượng công việc.
TerminateJobObject	Dừng tất cả tiến trình trong đối tượng công việc.
SetInformationJobObject	Thiết đặt những thông tin của đối tượng công việc.
QueryInformationJobObject	Lấy các thông tin của đối tượng công việc, như là thời gian dùng CPU, số tiến trình, danh sách định danh của tiến trình, hạn ngạch sử dụng, giới hạn bảo mật.

Ngoài ra, CPU và bộ nhớ cũng được giới hạn cho mỗi đối tượng công việc. Các giới hạn bao gồm:

- Giới hạn về số tiến trình đang hoạt động trong đối tượng công việc.
- Giới hạn về thời gian sử dụng CPU của mỗi tiến trình trong đối tượng công việc.

- Giới hạn về khoảng thời gian hoạt động của mỗi luồng trong từng tiến trình. Thông qua các lớp lịch chạy, cú cỏc khoảng thời gian sau, tuần tự cho mỗi luồng trong tiến trình.

- Mức ưu tiên cho tiến trình trong một đối tượng công việc: Mỗi tiến trình có mức độ ưu tiên riêng và nó khụng tự đặt được mức độ ưu tiên thông qua các hàm như SetThreadPriority.

- Giới hạn về bộ nhớ: định ra không gian địa chỉ ảo tối đa mà mỗi tiến trình trong một công việc được dùng.

Windows 2000 Datacenter Server có công cụ cho phép định nghĩa ra các công việc, đặt các hạn ngạch và giới hạn tài nguyên cho các tiến trình trong công việc. Đó là Process Control Manager.

Hai phương thức liên lạc giữa các tiến trình.

Liên lạc trực tiếp (Direct Communications)

§Theo địa chỉ đối xứng (Symmetric Scheme)

Send (P, Message) - Gửi thông điệp cho P

Receive (Q, Message) - Nhận thông điệp từ Q

Đặc điểm:

× Liên kết được thiết lập tự động giữa mỗi cặp tiến trình.

× Liên kết chỉ giữa 2 tiến trình.

× Chỉ có 1 liên kết giữa mỗi cặp.

× Tính đối xứng của liên lạc (2 bên đều biết đích xác tên của nhau khi Gửi/Nhận).

§Theo địa chỉ phi đối xứng (Asymmetric Scheme)

Send (P, Message) - Gửi thông điệp cho P

Receive (id, Message) - Nhận thông điệp từ tiến trình bất kỳ,
Biến id chứa số hiệu tiến trình gửi

Liên lạc gián tiếp (Indirect Communications)

§ Qua các Hộp thư (Mailboxes) hoặc Cổng (Ports).

§ Hộp thư là một thực thể qua đó thông điệp được gửi đến và lấy ra.

§ Mỗi hộp thư có định danh riêng.

§ Hai tiến trình phải chung nhau một hộp thư nào đó.

§ Hai loại hộp thư:

× Hộp thư tiến trình (Process Mailbox): Nằm trong vùng địa chỉ của một tiến trình nào đó.

× Hộp thư hệ điều hành (OS Mailbox): Nằm trong vùng địa chỉ của HĐH

Đồng bộ hoá liên lạc giữa các tiến trình.

Đồng bộ hoá liên lạc (Synchronization)

× Gửi thông điệp có chờ (Blocking Send)

× Gửi thông điệp không chờ (Nonblocking Send)

× Nhận thông điệp có chờ (Blocking Receive)

× Nhận thông điệp không chờ (Nonblocking Receive)

Trình bày mô hình chuyển trạng thái của tiến trình

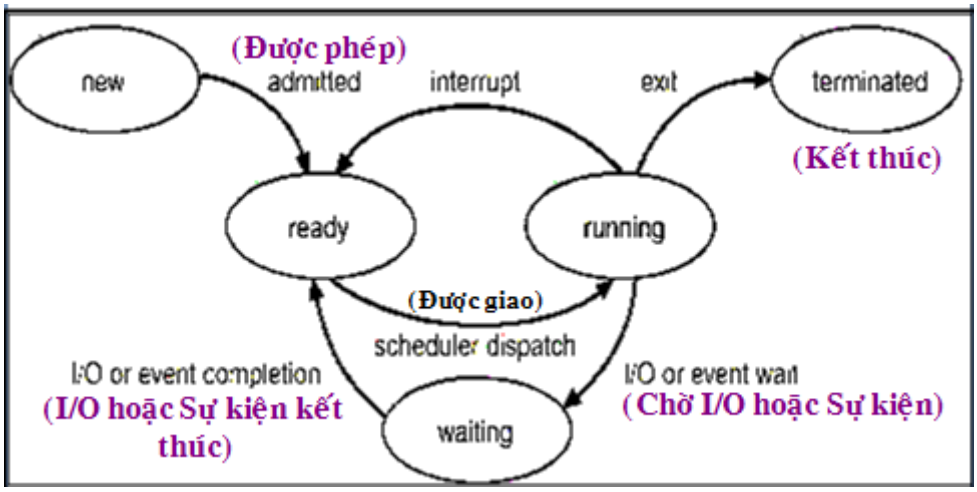
New: tiến trình đang được tạo lập.

Running: các chỉ thị của tiến trình đang được xử lý.

Blocked: 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.



Hình 5.1: Mô hình chuyển trạng thái của tiến trình

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 lặp lại cho P2 đến 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 biểu 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

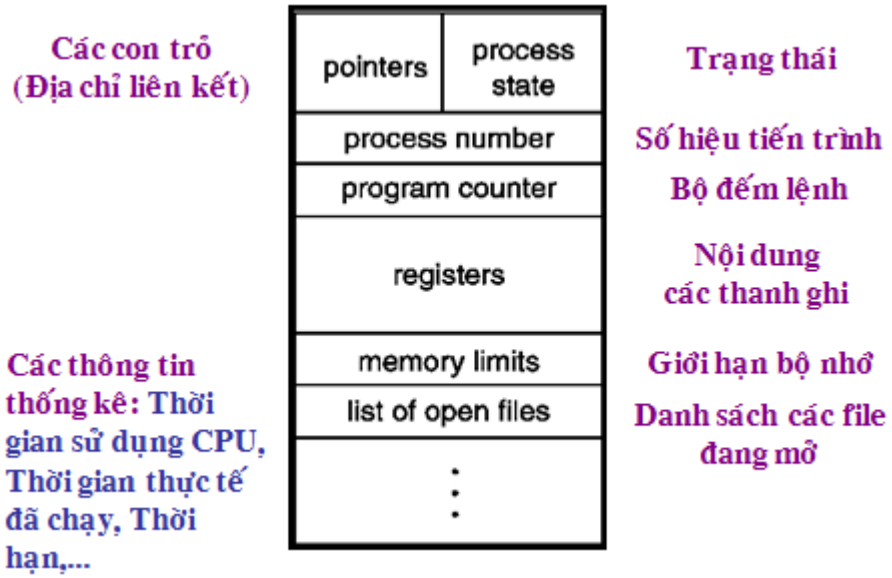
• Tiến trình yêu cầu một tài nguyên nhưng chưa được đáp ứng vì tài nguyên chưa sẵn sàng để cấp phát tại thời điểm đó ; hoặc tiến trình phải chờ một sự kiện hay thao tácnhập/xuất.

• 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.

6. 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:



Hình 6.1: 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…

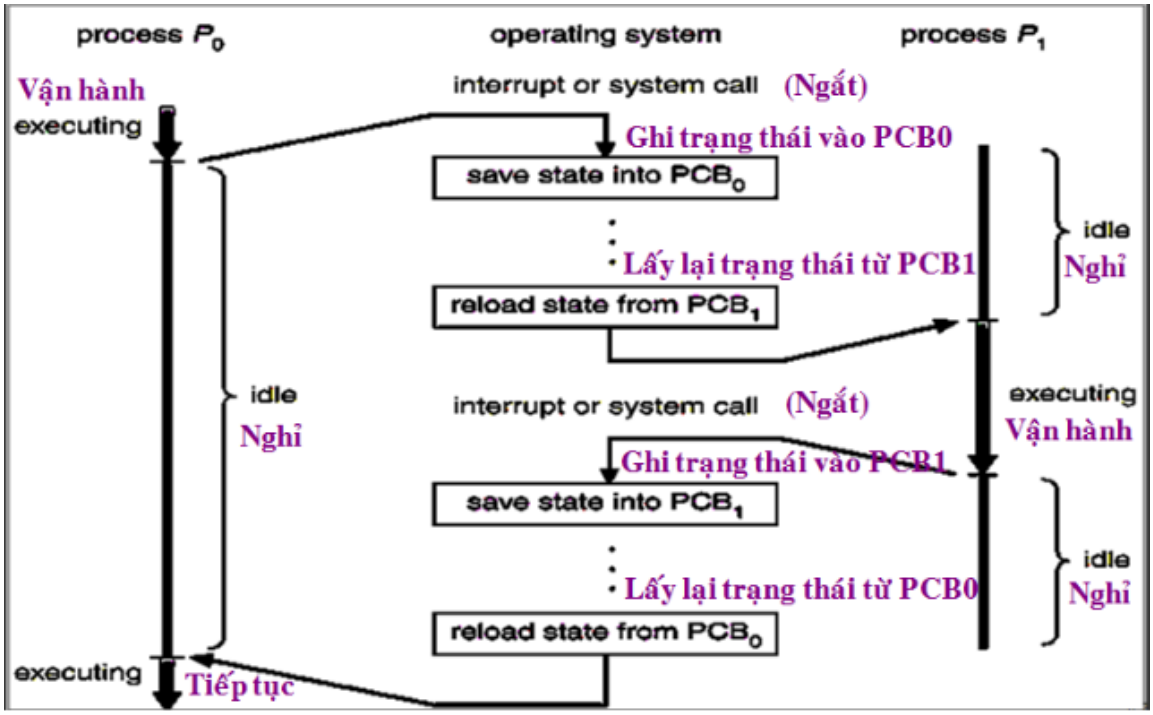
• Thông tin thống kê: CPU time, time limits…

• 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.

Trình bày mô hình luân chuyển CPU giữa hai tiến trình



Hình 6.2: 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.

7. Trình bày những lý do công tác giữa các tiến trình

Chia sẻ thông tin (Information Sharing): Một tiến trình sử dụng thông tin do tiến trình khác cung cấp. Tăng tốc tính toán (Computation Speedup): Các tiến trình cùng làm việc song song trên 1 hoặc nhiều máy để giải quyết bài toán chung.

Đảm bảo tính đơn thể (Modularity): Chương trình được chia thành các đơn thể chức năng vận hành trong các tiến trình hoặc luồng khác nhau. Ví dụ: mỗi bạn học một bài, đảm bảo tính đơn thể.

Đảm bảo tính tiện dụng (Convenience): Người dùng có nhu cầu làm nhiều việc một lúc: Soạn thảo, In ấn, Duyệt Web, Lấy file về, Biên dịch chương trình, Kiểm tra chính tả,...

Những lý do đồng bộ hóa công việc tiến trình.

- Đảm bảo tính nhất quán của tài nguyên dùng chung.
- Tránh được hiện tượng Deadlock (Hiện tượng kẹt tiến trình).
- Tính Loại trừ lẫn nhau hay Loại trừ tương hỗ (Mutual Exclusion) về phương diện thời gian: Khi có 1 tiến trình đang ở trong ĐTT của nó thì không có tiến trình nào khác trong nhóm cũng tại đoạn như vậy, nghĩa là: Mỗi thời điểm chỉ có 1 tiến trình được phép truy cập và/hoặc thay đổi tài nguyên chung.

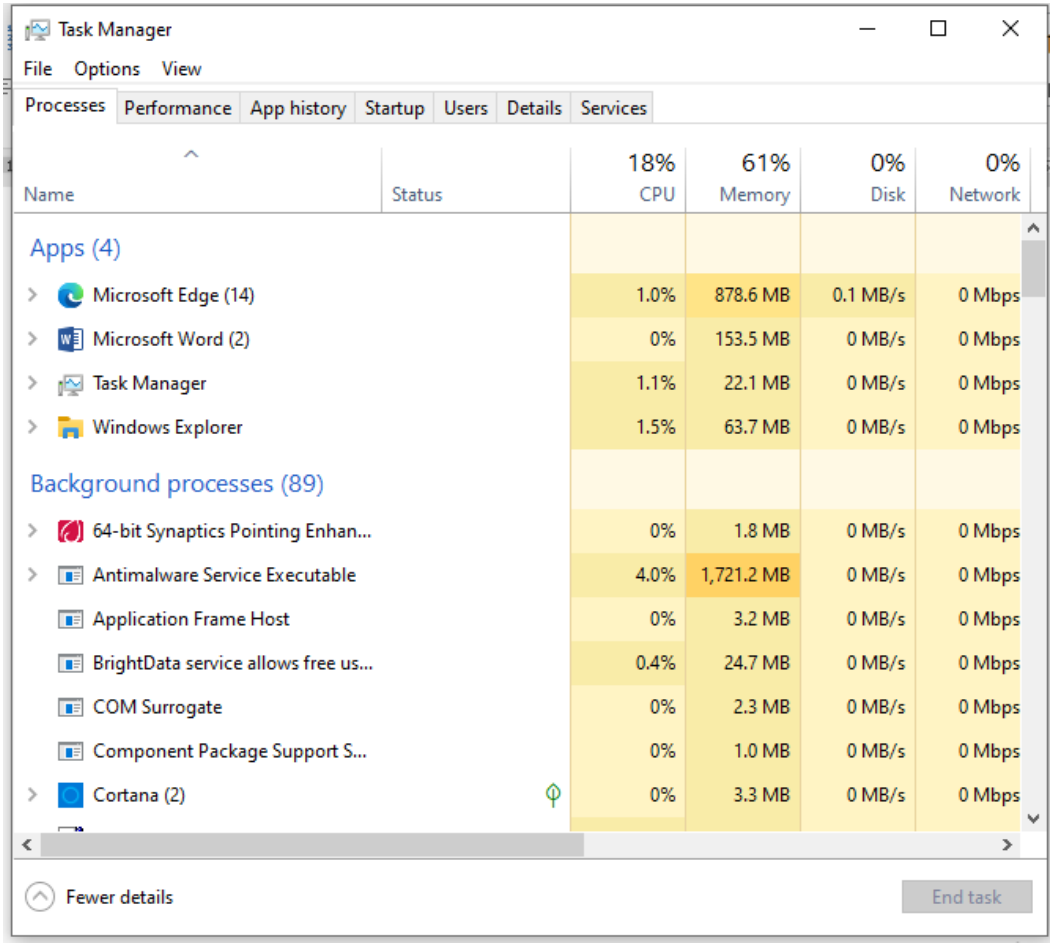
8. Quản lý tiến trình của hệ điều hành windows thông qua Task Manager

a. Khái niệm Task Manager

Task Manager của Windows là một công cụ rất mạnh được đóng gói với những thông tin rất hữu ích, từ tổng tài nguyên dung lượng trong hệ thống của bạn cho tới những chỉ số cụ thể nhất của mỗi tiến trình.

Các chương trình đang mở trên hệ thống sẽ được liệt kê tại đây, và tất nhiên cũng bao gồm cả các chương trình khởi động cùng hệ thống đang chạy trên nền background.

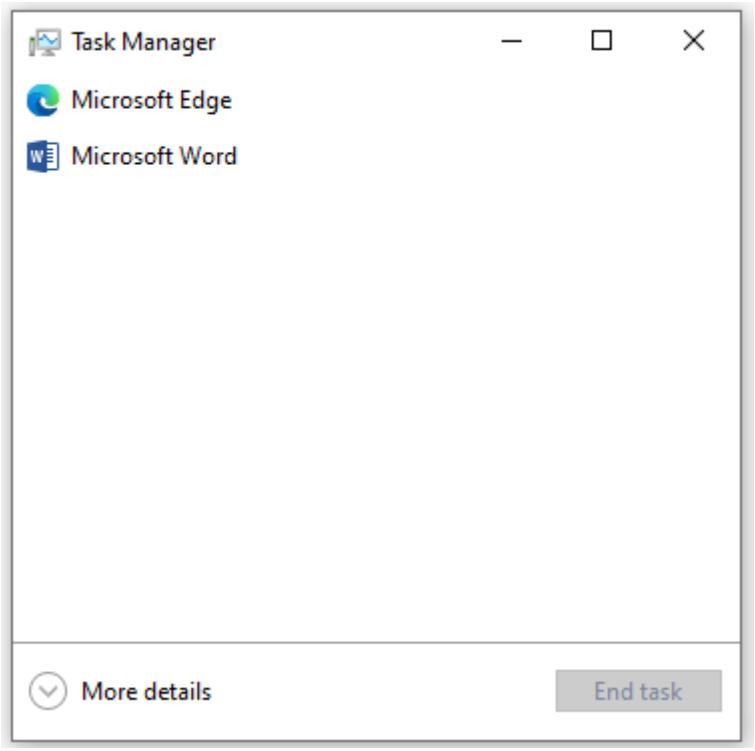
Bạn có thể sử dụng Task Manager để buộc kết thúc bất kỳ một chương trình, phần mềm nào đang chạy, cũng như để xem các chương trình "ngốn" bao nhiêu tài nguyên phần cứng trên máy tính của bạn, những chương trình và dịch vụ nào khởi động cùng hệ thống, … .



Hình 8.1: Màn hình giao diện của Task Manager

b. Quản lý tiến trình của hệ điều hành Windows thông qua Task Manager

Thanh quản lý tiến trình của Task Manager sẽ hiển thị danh sách bao quát các tiến trình đang chạy trên hệ thống của bạn. Nếu bạn sắp xếp các tiến trình đó theo tên, danh sách đó sẽ được chỉa ra làm 3 mục. Nhóm "Apps" sẽ hiện ra danh sách của các phần mềm đang chạy mà bạn sẽ thấy giống như khi mà bạn click vào "Fewer details" ở giao diện chính để hiển thị danh sách rút gọn của Task Manager. Hai mục còn lại sẽ là các tiến trình bên trong hệ điều hành và background, và chúng sẽ hiển thị các tiến trình mà ta không thể thấy ở danh sách rút gọn.



Hình 8.2: Danh sách rút gọn của Task Manager

9. Phân biệt tiến trình tiền cảnh và hậu cảnh (Foreground & Background)

Tiến trình mà yêu cầu một người dùng khởi động nó hoặc tương tác với nó thì được gọi là tiến trình tiền cảnh (Foreground processes). Tiến trình mà chạy độc lập mà không cần tới tác nhân là người dùng, chạy ngầm trong hệ thống, không có giao diện hiển thị trên màn hình thì được gọi là tiến trình hậu cảnh (Background processes). Các chương trình và dòng lệnh của tiến trình tiền cảnh được chạy một cách tự động. Để thực hiện một lệnh chạy "hậu cảnh", hãy thêm dấu "&" sau câu lệnh.

Controlling processes		
(part of) command	**Meaning**	**Giải thích**
regular_command	Runs this command in the foreground.	Chạy lệnh ở chế độ tiền cảnh.
command &	Run this command in the background (release the terminal)	Chạy lệnh ở chế độ hậu cảnh.
jobs	Show commands running in the background.	Hiển thị các tiến trình hậu cảnh.
Ctrl+Z	Suspend (stop, but not quit) a process running in the foreground (suspend).	Tạm dừng một tiến trình tiền cảnh.
Ctrl+C	Interrupt (terminate and quit) a process running in the foreground.	Ngắt một tiến trình tiền cảnh.
%n	Every process running in the background gets a number assigned to it. By using the % expression a job can be referred to using its number, for instance fg %2.	Mỗi 1 tiến trình hậu cảnh được cấp một số gọi là job_number. Bằng cách sử dụng cú pháp &jobnumber trong lệnh fg để chuyển tiến trình này về tiền cảnh. Ví dụ: fg %jobnumber
bg [job_number]	Reactivate a suspended program in the background.	Chuyển một tiến trình đang chạy tiền cảnh sang hậu cảnh.
fg [job_number]	Puts the job back in the foreground.	Chuyển một tiến trình đang chạy hậu cảnh sang tiền cảnh.
kill	End a process (also see Shell Builtin Commands in the Info pages of bash)	Ngắt tiến trình.

Hình 9.1: Bảng điều khiển các tiến trình

KẾT LUẬN

Tổng kết lại các nội dung được trình bày trong tài liệu này, ta đã nắm được quản lý tiến trình trong hệ điều hành Windows. Từ đó, chúng ta có thể hiểu được một phần cách vận hành, hoạt động hệ thống của hệ điều hành cũng như các tiến trình được thực hiện như thế nào. Tuy rằng tài liệu chỉ mang đến một phần kiến thức tuy nhỏ nhưng với chúng em thì cũng khá đầy đủ và căn bản để có thể tiếp tục nghiên cứu sâu hơn trong vấn đề quản lý tiến trình trong hệ điều hành Windows. Những nội dung chính như khái niệm chung về hệ điều hành Windows, khái niệm tiến trình, quá trình tạo một tiến trình, khái niệm luồng, cách kiểm tra hoạt động của một luồng, đối tượng công việc, vai trò của khối kiểm soát tiến trình, những lí do công tác giữa các tiến trình, khái niệm Task Manager và sự khác nhau giữa tiến trình tiền cảnh và hậu cảnh đều được trình bài trong tài liệu. Thông qua phần bài tập lớn này chúng em đã thấy được tầm quan trọng của việc quản lý các tiền trình bên trong một hệ thống.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Thanh Hải - Giáo Trình Nguyên Lý Hệ Điều Hành Đại học Công nghiệp Hà Nội, khoa công nghệ thông tin, Hà Nội 2016.
[2] Nguyễn Kim Tuấn - Giáo Trình Lý Thuyết Hệ Điều Hành Đại học Huế, TrườngĐại học khoa học, khoa công nghê thông tin, Huế 06/2004.
[3] Trần Hồ Thủy Tiên - Giáo Trình Nguyên Lý Hệ Điều Hành Đại học Đà Nẵng, trường Đại học Bách Khoa, Khoa Công Nghệ Thông Tin 01/04/2010.
[4] Abraham Silberschatz, Galvin, Gagne, Operating System Concepts 8th edition. (tài liệu tham khảo từ nguồn nước ngoài).
[5] v1.0010110225 (topica.edu.vn)
[6] Task manager: manage processes in Windows (getfastanswer.com)
[7] Tiến trình tiền cảnh & hậu cảnh (Foreground & Background) - Network Technology (google.com)
[8] Processes - IBM Documentation