ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

LUẬN VĂN TỐT NGHIỆP

PHÁT TRIỂN
HỆ THỐNG BE-PUM

Giáo Viên Hướng Dẫn:
PGS.TS. QUẢN THÀNH THƠ
ThS. NGUYỄN MINH HẢI
ThS. LÊ ĐÌNH THUẬN

Sinh viên thực hiện:
NGUYỄN XUÂN KHÁNH - MSSV: 51101594
NGUYỄN LÂM HOÀNG YÊN - MSSV: 51104402

Giáo Viên Phản Biện:
TS. BÙI HOÀI THẮNG

TP.Hồ Chí Minh, Ngày 10 tháng 12 năm 2015

LỜI CAM KẾT

Chúng tôi xin cam đoan rằng mọi thông tin và công việc được trình bày trong bài báo cáo này ngoài việc tham khảo các nguồn tài liệu khác có ghi đầy đủ trong phần phụ lục các tài liệu tham khảo, thì đều do chính chúng tôi thực hiện và chưa có phần nội dung nào được sao chép từ các đề tài thực tập tốt nghiệp, luận văn tốt nghiệp của trường này và trường khác. Nếu có bất kì sai phạm hay gian lận nào, chúng tôi xin chịu hoàn toàn trách nhiệm trước Ban Chủ Nhiệm Khoa và Ban Giám Hiệu Nhà Trường.

TP. Hồ Chí Minh, Ngày 10 tháng 12 năm 2015

Nhóm sinh viên thực hiện đề tài

LỜI CẢM ƠN

Đầu tiên em xin gửi lời cảm ơn sâu sắc đến thầy PGS.TS. Quản Thành Thơ, thầy đã có những sự định hướng, động viên, giải đáp và quan tâm to lớn đến em và cho cả mọi người đang cùng dưới sự dẫn dắt của thầy. Thầy luôn theo sát mọi người qua từng ngày, từng tuần thực hiện nghiên cứu và đưa ra những gợi ý để công việc được suôn sẻ cũng như đạt kết quả như mong muốn. Những lời chỉ bảo và răn đe của thầy luôn là động lực để em và mọi người hoàn thành nhiệm vụ. Chính phong thái đó của thầy là lý do để em chọn vào tham gia nghiên cứu trong nhóm của thầy; và càng ngày em càng cảm ơn về sự lựa chọn đúng đắn đó.

Kế đến em xin dành sự biết ơn to lớn dành cho ThS. Nguyễn Minh Hải, với em anh không chỉ là một người hướng dẫn đề tài, mà còn là một người anh gần gũi vì những sự trợ giúp ân cần từ những ngày đầu em tham gia cùng mọi người thực hiện nghiên cứu. Làm việc với anh không phải là một sự cứng nhắc và rập khuôn, bởi anh luôn thân thiện thảo luận cùng mọi người để đưa ra những ý kiến, công việc và thời hạn hợp tình, hợp lý. Anh luôn dành sự quan tâm và tận tình cho từng thành viên dưới sự hướng dẫn của anh; kèm theo đó là những lời động viên, chia sẻ để từ đó em có thêm động lực để hoàn thành được đề tài này.

Em cũng xin cảm ơn ThS. Lê Đình Thuận đã giúp em hoàn thiện bài báo cáo này qua việc theo dõi sát sao mỗi ngày trong thời gian viết bài để hướng dẫn, đóng góp ý kiến và giúp em sửa chữa những sai sót có trong báo cáo.

Em xin dành những sự tri ân chân thành và to lớn đến toàn thể quý thầy cô của khoa Khoa học và Kỹ thuật Máy tính mà em đang theo học. Nhờ những sự tận tình giảng dạy, truyền đạt kiến thức và kinh nghiệm quý báu qua từng bài giảng mà em có thêm được sự vun đắp, vững chắc trong vốn hiểu biết của mình; để từ đó làm nền tảng cho em hoàn thành được đề tài ngày hôm nay.

Em cũng không quên bày tỏ lòng biết ơn đến gia đình, người thân của mình, những người đã luôn quan tâm, động viên và chăm sóc em qua từng ngày để em có được sức khỏe, tinh thần và môi trường phát triển, giúp em có được như ngày hôm nay. Cuối cùng em xin cảm ơn những người bạn đã hỗ trợ, chia sẻ để em hoàn thành tốt được đề tài này.

Với toàn bộ sự biết ơn sâu sắc đó, em xin gửi lời chúc đến mọi người đã giúp em làm được những điều đến ngày hôm nay. Chúc cho mọi quý thầy cô, đặc biệt là những người em đã nêu tên ở trên, của khoa Khoa học và Kỹ thuật Máy tính luôn có được một sức khỏe dồi dào để luôn hạnh phúc trong cuộc sống và có khả năng để cống hiến thêm cho khoa, cho trường và cho thế hệ đi sau.

Trân trọng.

TP. Hồ Chí Minh, 19/12/2015

Nhóm sinh viên thực hiện đề tài

TÓM TẮT LUẬN VĂN

Mục lục

Danh sách hình ảnh

Danh sách các bảng

Chương 1

Giới Thiệu

1.1 Giới thiệu sơ lược về BE-PUM

BE-PUM tên đầy đủ là Binary Emulation for Pushdown Model generation, là một công cụ dùng để phân tích động mã nhị phân của một chương trình bất kỳ chạy trên kiến trúc X86 của hệ điều hành Microsoft Windows nền tảng 32-bit. Sau khi phân tích, BE-PUM sẽ sinh ra hợp ngữ – mã assembly và đồ thị luồng điều khiển (control flow graph – CFG) của chương trình đầu vào.

BE-PUM được xây dựng chính trên mã nguồn của JakStab nhưng không hạn hẹp ở việc chỉ phân tích tĩnh, BE-PUM có thể phân tích động và chỉ ra lại mỗi dòng lệnh của mã assembly môi trường làm việc của nó là như thế nào. Việc này sẽ giải quyết được những trường hợp phân tính vào những nhánh không cần thiết – không bao giờ được thực thi hoặc khi chương trình đang cố gắng thay đổi chính nội dung của mình.

Với việc phân tích mã nhị phân đó, BE-PUM đang được phát triển để tập trung vào phân tích những phần mềm bị nghi ngờ để rồi sau đó sẽ phát hiện được những kỹ thuật tấn công, và cuối cùng là xác định xem đây có thực là phần mềm gây hại đến máy tính hay không?

1.2 Tổng quan về Windows API

Với tên gọi đầy đủ là Microsoft Windows application programming interface, đôi lúc được gọi một cách ngắn gọn là WinAPI, đây là một bộ giao diện lập trình ứng dụng (API) lõi có sẵn trong hệ điều hành Microsoft Windows (hay thường được gọi ngắn gọn là Windows).

Windows API là tên gọi chung của những dự án được triển khai cho những nền tảng khác nhau được áp dụng trong hệ điều hành Windows; mà thông thường chúng có những tên gọi riêng. Một số tên gọi và mô tả cho những phiên bản xin được trình bày trong Bảng 1.1.

Tên của phiên bản Windows API	Mô tả
Win16	Đây là tên gọi bộ API đầu tiên có trong phiên bản Microsoft Windows 16-bit. Thuở ban đầu bộ API này có tên gọi là "Windows API", nhưng sau khi phiên bản kế tiếp là Win32 ra đời, nó được đổi tên thành Win16. Lúc này các chức năng của Win16 API chủ yếu nằm trong các tập tin lõi của hệ điều hành như kernel.exe (hoặc krnl286.exe hoặc krnl386.exe), user.exe và gdi.exe. Dù rằng phần mở rộng là exe, nhưng thực tế đây là các thư viện liên kết động.
Win32	Đây là tên gọi cho bộ Windows API trong phiên bản Windows 32-bit, được giới thiệu lần đầu cùng hệ điều hành Windows NT. Các chức năng của Win32 API cũng bao gồm cả những chức năng của Win16 API, nhưng khác với phiên bản trước, chúng được đóng gói trong các tập tin DLL; trong đó có thể kể đến những tập tin DLL cốt lõi của Win32 API như: kernel32.dll, user32.dll, và gdi32.dll.

Win64	Đây là một biến thể sau đó của bộ API được hiện thực trong nền tảng Windows 64-bit. Cả chương trình 32-bit hoặc 64-bit đều có thể được biên dịch với cùng một mã nền duy nhất, dù rằng một số API trong Win32 đã được thay thế và loại bỏ. Tất cả con trỏ trong phiên bản này mặc định là 64-bit, do đó cần kiểm tra khả năng tương thích trong mã nguồn và viết lại nếu cần thiết.

Bảng 1.1: Một số phiên bản chính của Windows API

1.3 Tổng quan về assembly

Ngôn ngữ assembly (hay hợp ngữ) là ngôn ngữ cấp thấp được biên dịch để thực thi một chương trình nào đó. Ngôn ngữ asseambly có tính gợi nhớ, mỗi câu lệnh thực thi một chức năng, hay nhiệm vụ riêng biệt tưng ứng với một lệnh yêu cầu máy thực thi.

Các chương trình sau khi biên dịch ra thành nhiều dạng file khác nhau để thực thi. Mỗi chương trình được biên dịch thành ngôn ngữ asembly trước khi biên dịch qua ngôn ngữ máy để máy hiểu và thực thi câu lênh. Khi biên dịch qua mã assembly, đọc đoạn mã assembly để phân tích xem chương trình này có phải là virus máy tính không, có những hành vi bất thường từ đó xem xét đưa ra kết quả chương trình này có an toàn với máy tính.

1.4 Mục tiêu đề tài

Trong phạm vi của đề tài thực tập tốt nghiệp, mục tiêu nhắm tới là phát triển hệ thống xử lý các Windows API cho BE-PUM. Với số lượng các API rất lớn hiện có trong hệ điều hành Windows, hiện tại đề tài đang tập trung vào xử lý các API ở phiên bản Win32 API, do hầu hết các phần mềm độc hại mà BE-PUM hướng tới vẫn đang dùng bộ API này; với sự ưu tiên từng bước xây dựng cho các API được dùng phổ biến trước.

Bên cạnh việc nhận thông tin đầu vào từ vùng nhớ đã được xây dựng của BE-PUM và trả về kết quả sau khi gọi API vào đúng địa chỉ cần thiết, điều quan trọng là phải đảm bảo không gây ngắt quãng cũng như tránh nguy hại hệ thống đang chạy. Và như vậy với những tương tác vật lý từ lời gọi API (bộ lưu trữ máy tính, cơ sở dữ liệu registry...) hay tương tác người dùng (API tạo cửa sổ message box, lệnh cho một thread "ngủ đông" trong một khoảng thời gian,...) cần được kiểm soát để không làm ảnh hưởng tới kết quả thực thi của BE-PUM.

Lưu ý: do nội dung đề tài tập trung vào xử lý cho Win32 API, nên kể từ đây, khi báo cáo nhắc đến Windows API tức là nói đến Win32 API.

Trong phạm vi của đề tài luận văn tốt nghiệp, mục tiêu nhắm tới là hiện thực câu lệnh assembly với số lượng lớn nhất có thể, hiện số lượng câu lệnh assembly đã được hiện thực lên tới 200 câu lệnh xử lý xử lý số nguyên, 50 câu lệnh xử lý số thập phân, đã góp phần phân tích một số loại virus đang nghiên cứu, các câu lệnh đã được tiến hành kiểm tra. Dựa trên một số hành virus hiện nay và những câu lệnh phổ biến thường gặp, BE-PUM đã hỗ trợ tốt công việc phân tích đúng môi trường làm việc của từng câu lệnh từ đó xác định chính xác và phân tính đúng các kỹ thuật mà virus đang tiến hành.

Bên cạnh đó, BE-PUM còn xây dựng chương trình so trùng đồ thị, với những đồ thị mẫu được cho là của chương trình độc ai, BE-PUM trả kết quả là vị trí đỉnh đầu tiên mà chương trình đó bắt đầu, công việc này có thể được xem là tách chương trình độc hại từ chương trình cần phân tích. Quá trình xây dựng so trùng đồ thị chỉ ở mới bước bắt đầu thực hiện, mục tiêu đặt ra là trả về đồ thị được cho là chương trình an toàn.

1.5 Cấu trúc của báo cáo

Bài báo cáo này bao gồm những đề mục sau đây:

Chương 1

Giới thiệu tổng quan về BE-PUM, yếu tố quyết định để cho ra đề tài này; dẫn nhập về Windows API, thành phần sẽ được áp dụng để phát triển cho BE-PUM; dẫn nhập về hợp ngữ assembly, và cuối cùng nêu ra được mục đính chính của đề tài sẽ cần làm gì.

Chương 2

Đem đến những cái nhìn về những vấn đề đã và đang được lưu tâm khi thực hiện đề tài này; sự phổ biến của Windows API trong những phần mềm độc hại để thấy sự cần thiết của việc xây dựng một bộ xử lý Windows API cho BE-PUM; những khó khăn khi thực hiện điều đó và giải pháp cho vấn đề. Đông thời phân tích vấn đề đặt ra là hiện thực các câu lệnh trong hợp ngữ assembly, giải thích tại sao sử dụng ngôn ngữ assembly để tiến hành phân tích chương trình.

Chương 3

Trình bày những kiến thức cần thiết cho quá trình thực hiện đề tài; từ những kiến thức phải nắm được về hệ thống BE-PUM do đây là một đề tài làm việc dựa trên đó; và mỗi khi làm việc với một thư viện bất kỳ, đòi hỏi ta phải tìm hiểu cách thức làm việc với thư viện đó và cả những kiến thức cần thiết do bộ thư viện ấy yêu cầu. Đồng thời cũng trình bày các kiến thức cơ bản về hợp ngữ assembly, từ đó có cái nhìn tổng quan về assembly để tiến hành xây dựng chương trình BE-PUM.

Chương 4

Mỗi chương trình bất kỳ đều cần một thiết kế tốt để giúp cho việc xây dựng dễ dàng và quy chuẩn hơn. Mục này sẽ trình bày cách mà bộ xử lý Windows API đã được hiện thực để tương tai sau này có thể dễ dàng sửa chữa, bảo trì và bổ sung thêm vào kiến trúc đó. Để hiểu rõ cấu chương trình mô phỏng câu lệnh assembly, sơ đồ class trình bày trong chương này thể hiện được mối quan hệ giữa các class, cấu trúc của chương trình BE-PUM được phát triển dựa trên dự án JakStab, giới thiệu các class quan trọng của chương trình BE-PUM

Chương 5

Trình bày về kết quả mà bộ xử Windows API đã đạt được với những Windows API đã được hỗ trợ cho hệ thống BE-PUM. Đồng thời trình bày các bước để hiện thực một câu lệnh của hợp ngữ assembly, cách đánh giá kết quả đúng hay sai bằng cách so sánh kết quả với chương trình OnlyDbg, giới thiệu về công cụ MASM.

Chương 6

Liệt kê về những tài liệu và nguồn tham khảo có liên quan đến đề tài này.

Chương 2

Phân Tích Vấn Đề

2.1 BE-PUM và những khó khăn

2.1.1 Mục tiêu hướng đến của BE-PUM

2.1.2 Các hướng tiếp cận

2.1.3 Mô hình luồng điều khiển

2.1.4 Những đòi hỏi trong quá trình hiện thực

2.2 Các câu lệnh hợp ngữ

2.2.1 Assembly ngôn ngữ dùng để phân tích

2.2.2 BE-PUM và Assembly

2.2.3 Phân tích Assembly trong BE-PUM

2.3 Windows API

2.3.1 Windows API trong những phần mềm độc hại

Để cung cấp sức mạnh và sự tiện lợi cho lập trình viên trong việc viết ứng dụng chạy trên hệ điều hành Windows, các API trong bộ Windows API mở ra nhiều cách thức nhanh

chóng và mạnh mẽ cho lập trình viên trong việc tương tác với hệ thống.

Và vấn đề gì cũng có hai mặt của nó, sự hỗ trợ mạnh mẽ đó cũng là con đường đơn giản để các tin tặc áp dụng vào việc xây dựng nên các phương pháp tấn công, cũng như cho ra đời những phần mềm nguy hại (malware), để lại bao hậu quả xấu cho hệ thống máy vi tính trên toàn cầu.

Trong quá trình xây dựng BE-PUM và qua việc phân tích hàng ngàn mẫu malware chạy trên môi trường Windows được phát tán ở khắp nơi trên thế giới, hầu hết những mẫu malware trên đều áp dụng lời gọi Windows API vào cách thức tấn công của chúng. Những phương pháp tấn công phổ biến như SEH hay phương pháp chống phát hiện đều có sự tồn tại của Windows API trong đó.

Do đó, việc xây dựng một bộ công cụ xử lý những thông tin trả về từ Windows API là rất cần thiết cho việc phát triển hệ thống BE-PUM, một hệ thống tập trung vào phân tích mã nhị phân của malware.

2.3.2 BE-PUM và Windows API

Mã nguồn của những API trong bộ Windows API được tập đoàn Microsoft giữ kín và không hề công bố. Chỉ có những đặc tả và hướng dẫn sử dụng được Microsoft phổ biến rộng rãi cho lập trình viên. Nghĩa rằng ta chỉ có thể biết được đầu vào của lời gọi và mong muốn đầu ra sẽ như ý, chứ không thể nắm rõ lô-gíc xử lý bên trong của chúng. Điều đó khiến cho việc xử lý đúng đắn một cách tổng quát đối với mọi đầu vào của mỗi API bằng cách viết lại bộ mã xử lý tương ứng của chúng vào trong BE-PUM dường như trở nên không thể.

Hướng tiếp cận hiện tại là tiến hành lấy nội dung bộ nhớ, nội dung các đối số nằm trên stack bên trong BE-PUM và tiến hành gọi thực sự với Windows API, nhận kết quả

trả về và nạp lại vào trong BE-PUM để tiếp tục tiến hành phân tích các câu lệnh tiếp theo.

BE-PUM là một dự án được phát triển lên từ nhân của dự án JakStab và được viết hoàn toàn trên ngôn ngữ lập trình Java. Với Windows API thì lại là một câu chuyện hoàn toàn khác, Windows API được phát triển chủ yếu tập trung vào ngôn ngữ lập trình C kèm với các mô tả và cấu trúc dữ liệu được viết trên đó. Thêm lần nữa, việc hiện thực ý tưởng gọi để lấy kết quả Windows API từ trong lòng BE-PUM gặp nhiều khó khăn. Đặc biệt là việc ánh xạ các dữ liệu kiểu cấu trúc giữa hai thành phần trên cũng là một trăn trở.

Vì những lý do trên, cần tìm hiểu một cách thức giải quyết vấn đề nhanh chóng và đơn giản hơn bằng một bộ công cụ nào đó để xử lý rào cản ngôn ngữ giữa Java và C. Thêm vào đó, bộ công cụ này cũng cần có tính linh hoạt và mềm dẻo để cho việc phát triển về sau được dễ dàng.

2.3.3 Truy xuất Windows API bên trong BE-PUM thông qua JNA

Vấn đề trên được giải quyết thông qua bộ thư viện Java Native Access (JNA).

Java Native Access là một thư viện được cộng đồng phát triển, nhằm giúp cho các chương trình được viết bằng ngôn ngữ lập trình Java dễ dàng truy cập vào các thư viện native shared mà không cần thông qua Java Native Interface. Thiết kế của JNA cũng cung cấp khả năng này mà không cần bỏ ra nhiều công sức.

Với khả năng ánh xạ dễ dàng giao diện lập trình giữa hai ngôn ngữ Java và C; bao gồm ánh xạ tên hàm, kiểu dữ liệu trả về, kiểu dữ liệu của các thông số đầu vào; từ những kiểu dữ liệu cơ bản đến những kiểu dữ liệu cấu trúc và kể cả con trỏ; đó là những ưu điểm để lựa chọn JNA áp dụng vào trong việc giải quyết yêu cầu của đề tài nêu trên.

Chương 3

Kiến Thức Nền

3.1 Làm việc với BE-PUM

3.2 Các câu lệnh hợp ngữ

3.2.1 Assembly

3.2.1.1 Bộ nhớ assembly

3.2.1.2 Câu lệnh assembly

3.2.1.3 Tập giá trị

3.2.1.4 Kiểu giá trị

3.2.1.5 Cấu trúc một chương trình assembly

3.2.2 Floating-Point Unit (FPU)

3.2.2.1 Số thực và định dạng Floating-point

3.2.2.2 Kiến trúc FPU

3.2.2.3 Loại dữ liệu dấu chấm động và định dạng

3.2.2.4 Câu lệnh FPU

3.3 Windows API

3.3.1 Một vài bộ thư viện cần hỗ trợ

Ở những bước xây dựng đầu của đề tài, cần ưu tiên tiến hành cho những bộ thư viện phổ

Tên	Chức năng
Dịch vụ nền	Cho phép truy cập vào các nguồn tài nguyên cơ bản có sẵn trong hệ thống của Windows. Bao gồm những thứ như hệ thống tập tin, thiết bị, tiến trình (process), luồng (thread), xử lý lỗi (error handing). Các chức năng này nằm trong tập tin kernel32.dll trên hệ điều hành Windows 32-bit.
Dịch vụ nâng cao	Cho phép truy cập vào các chức năng bổ sung. Bao gồm những thứ như Windows registry, tắt/khởi động lại hệ thống (hoặc bãi bỏ - abort), bắt đầu/dừng/tạo ra một dịch vụ, quản lý tài khoản người dùng. Các chức năng này nằm trong tập tin advapi32.dll trên hệ điều hành Windows 32-bit.
Giao diện người dùng	Cung cấp các chức năng để tạo và quản lý cửa sổ màn hình và hầu hết những điều khiển cơ bản, chẳng hạn như nút và thanh cuộn, nhận chuột và bàn phím, cũng như các chức năng khác liên quan đến phần giao diện đồ họa người dùng của Windows.
Các chức năng này nằm trong tập tin user32.dll trên hệ điều hành Windows 32-bit. Windows shell	Cho phép các ứng dụng truy cập các chức năng được cung cấp bởi các shell của hệ điều hành, cũng như thay đổi nó. Windows shell ở đây được hiểu là những thành phần cấu tạo nên giao diện đồ họa người dùng của hệ điều hành Windows (bao gồm những thứ như: màn hình desktop, thanh làm việc taskbar và các thành phần con trên nó,...). Các chức năng này nằm trong tập tin shell32.dll trên hệ điều hành Windows 32-bit.

Bảng 3.1: Những bộ thư viện được hỗ trợ bởi BE-PUM

3.3.2 Trình tự các bước để ứng dụng JNA vào BE-PUM

Để tiến hành áp dụng những khả năng mang lại từ JNA và hệ thống BE-PUM, ta cần trải qua những bước sau đây:

1. Ánh xạ tương ứng tên thư viện sẽ được gọi

2. Ánh xạ tên hàm của API và những kiểu dữ liệu sẽ được dùng (bao gồm kiểu dữ liệu trả về và kiểu dữ liệu của các thông số đầu vào) từ ngôn ngữ lập trình C sang Java
3. Tiến hành lấy giá trị bộ nhớ của các thông số đầu vào có trong BE-PUM
4. Truyền các giá trị đó vào những kiểu tương ứng đã được ánh xạ trong Java, gọi API và lấy kết quả trả về.
5. Lưu giá trị nhận được đó về lại bộ nhớ của BE-PUM

3.3.3 Những kiểu dữ liệu và ánh xạ của chúng vào JNA

Việc ánh xạ kiểu dữ liệu từ ngôn ngữ lập trình C sang Java được hỗ trợ sẵn bởi JNA cho một số kiểu cơ bản thường dùng như sau:

STT	Kiểu dữ liệu trong C	Kiểu dữ liệu trong Java
1	char	byte
2	wchar_t	char
3	short	short
4	int	int
5	int	boolean
6	enum	int (thông thường)
7	long long, __int64	long
8	float	float
9	double	double
10	Con trỏ (VD: void*)	Buffer Pointer
11	Con trỏ (VD: void*, char*) Mảng	<P>[] (mảng của những kiểu nguyên thủy)
12	long	NativeLong
13	const char*	String
14	const wchar_t*	WString
15	char**	String[]

16	wchar_t**	WString[]
17	void**	Pointer[]
18	struct* struct	Structure
19	union	Union
20	struct[]	Structure[]
21	void (*FP)()	Callback
22	pointer (<T> *)	PointerType

Bảng 3.2: Một số ánh xạ kiểu dữ liệu cơ bản

Căn bản việc ánh xạ kiểu dữ liệu trong JNA được đáp ứng bằng việc kích thước dữ liệu được sử dụng giữa hai ngôn ngữ C và Java có kích thước bằng nhau. JNA hỗ trợ cho việc nạp một mảng các kí tự trong C bằng cách cho vào một đối tượng kiểu String.

Điểm hỗ trợ mạnh trong JNA đó là sử dụng và truy cập con trỏ của C thông qua đối tượng kiểu Pointer, dù rằng đang làm việc trong ngôn ngữ Java. Và nếu muốn lấy giá trị của một vùng nhớ thay vì chỉ một giá trị ô nhớ như Pointer, ta có thể sử dụng đối tượng kiểu Buffer để JNA đưa giá trị vùng nhớ đó vào trong đối tượng.

3.3.4 Dữ liệu kiểu cấu trúc (structure)

Một trong những lợi thế to lớn nhất mà JNA mang lại đó là nạp dữ liệu kiểu cấu trúc (structure) vào trong C mà ngôn ngữ Java không có. Điều này được hỗ trợ qua việc tạo ra một lớp mới, kế thừa từ lớp Structure của JNA.

Để tạo ra chính xác kiểu cấu trúc để nạp vào JNA, ta cũng phải tìm hiểu và nắm rõ các quy tắc xây dựng và làm việc với kiểu cấu trúc trong ngôn ngữ lập trình C.

Một struct trong ngôn ngữ lập trình C là một khai báo dữ liệu kiểu phức hợp, trong đó nó định nghĩa một danh sách các biến được đặt bên trong một khối bộ nhớ. Nó cho

phép các biến khác nhau được truy xuất thông qua một con trỏ duy nhất (con trỏ của structure). Một structure có khả năng chứa nhiều kiểu dữ liệu khác nhau từ kiểu dữ liệu nguyên thủy đến kiểu dữ liệu phức tạp khác (enum, structure).

Định nghĩa struct khá giống với định nghĩa lớp trong Java, nhưng struct không hề có khả năng khai báo phương thức, cũng như các từ khóa public, private,... như Java. Thêm vào đó, cơ chế cấp phát bộ nhớ của struct chỉ đơn thuần là sự sắp xếp liên tục của các vùng nhớ chứa giá trị của các biến đã được khai báo bên trong. Cách sắp xếp này hoàn toàn tương ứng với ngôn ngữ assembly; do đó ta cần đảm bảo việc lấy ra và gán trở lại bộ nhớ BE-PUM một cách chính xác theo trình tự như vậy.

Chương 4

Thiết Kế Và Xây Dựng

4.1 Các câu lệnh hợp ngữ

4.1.1 Sơ đồ chương trình BE-PUM

4.1.2 Phân tích class

4.2 Windows API

4.2.1 Các thành phần chính của bộ xử lý Windows API

4.2.2 Những khai báo ánh xạ của bộ xử lý Windows API

4.2.3 Kiến trúc xây dựng và làm việc

4.2.4 Quản lý môi trường tương tác vật lý

Chương 5

Kết Quả

5.1 Thí nghiệm và đánh giá kết quả

5.2 Các câu lệnh hợp ngữ đã được hỗ trợ

5.3 Các Windows API đã được hỗ trợ

Chương 6

Hướng Phát Triển Trong Tương Lai

Tài liệu tham khảo

1. Example