**Họ và tên: Phạm Tiến Thành Công**

**msv: 20010886**

**Lớp:CNTT4**

* **Trang 1**

**6. Mechanism: Limited Direct Execution**(Cơ chế: Thực thi trực tiếp có giới hạn)

* Để ảo hóa CPU, HĐH cần chia sẻ máy vật lý giữa các công việc khác nhau. Cách đơn giản nhất để làm điều này là chạy một quá trình tại một thời điểm, sau đó chạy một quá trình khác sau đó.Đầu tiên là hiệu suất: làm thế nào chúng ta có thể giảm thiểu chi phí hệ thống mà không cần thêm quá nhiều quyền kiểm soát đối với CPU? Thứ hai là kiểm soát: làm thế nào chúng ta có thể giữ cho các tiến trình hoạt động hiệu quả trong khi vẫn bảo vệ CPU?
* Mục tiêu của HĐH là ảo hóa CPU theo cách duy trì quyền kiểm soát hệ thống. Để đạt được điều này, hệ điều hành sẽ cần hỗ trợ cả phần cứng và hệ điều hành.

**6.1 Kỹ thuật cơ bản: Thực hiện trực tiếp hạn chế**

* **Trang 2**
* Các nhà phát triển đã nghĩ ra một cách để làm cho các chương trình chạy trơn tru hơn, đơn giản là chạy chúng trực tiếp trên CPU. Kỹ thuật này được gọi là thực thi trực tiếp hạn chế.

OS Program

Create entry for process list

Allocate memory for program

Load program into memory

Set up stack with argc/argv

Clear registers

Execute call main()

Run main()

Execute return from main

Free memory of process

Remove from process list

Figure 6.1: Direct Execution Protocol (Without Limits)

* Trang 2
* Hướng dẫn này chỉ ra cách triển khai một giao thức thực thi trực tiếp cơ bản. Nó hoạt động bằng cách gọi main () của chương trình và quay trở lại hạt nhân.
* Câu hỏi đặt ra là: nếu chúng tôi chạy một chương trình, làm cách nào để hệ điều hành đảm bảo rằng nó không làm bất cứ điều gì không nên làm trong khi vẫn chạy hiệu quả? Làm như vậy sẽ cho phép hệ thống ảo hóa CPU.

**6.2 Problem #1: Restricted Operations**

(**Vấn đề # 1: Hoạt động bị hạn chế)**

* Vì quá trình thực thi trực tiếp rất nhanh nên nó có thể chạy trơn tru trên CPU phần cứng. Tuy nhiên, nó cũng đưa ra một vấn đề nếu chương trình muốn thực hiện một số hoạt động bị hạn chế.
* **Trang 3**
* **THE CRUX: CÁCH THỰC HIỆN CÁC HOẠT ĐỘNG HẠN CHẾ**   
   Một quy trình phải có khả năng thực hiện I / O và các hoạt động bị hạn chế khác mà không làm mất quyền kiểm soát hoàn toàn đối với hệ thống.

/\***ASIDE: WHY SYSTEM CALLS LOOK LIKE PROCEDURE CALLS(BÊN TRONG: TẠI SAO CÁC CUỘC GỌI HỆ THỐNG NHÌN NHƯ THỦ TỤC CUỘC GỌI)**

* Tại sao một lệnh gọi đến một lệnh gọi hệ thống giống hệt như một lệnh gọi thủ tục trong C? Đó là bởi vì các lệnh gọi hệ thống bị ẩn bên trong một lệnh gọi thủ tục.Khi bạn gọi open (), thư viện thường đặt các đối số của lệnh gọi thủ tục vào một vị trí nổi tiếng. Sau đó, lệnh gọi thủ tục này thực hiện lệnh bẫy sau.Mã trong thư viện thực hiện lệnh gọi hệ thống sau khi giải nén bẫy trả về điều khiển cho chương trình đã khởi tạo lệnh gọi.\*/
* Một cách tiếp cận sẽ là cho phép các tiến trình chạy các hoạt động I / O của riêng chúng, nhưng không cho phép chúng cấp I / O cho đĩa. Làm như vậy sẽ ngăn cản sự phát triển của nhiều hệ thống mong muốn.
* Chúng tôi giới thiệu một chế độ xử lý mới được gọi là chế độ người dùng. Chế độ này giới hạn những gì một bộ xử lý có thể làm.
* Ngược lại với chế độ người dùng, chế độ hạt nhân là chế độ mà hệ điều hành có thể chạy trong các đặc quyền chính thức. Chế độ này có thể thực hiện nhiều loại hoạt động bị hạn chế khác nhau.Hầu hết phần cứng hiện đại có khả năng cho phép các chương trình người dùng thực hiện các cuộc gọi hệ thống. Các lệnh gọi này hiển thị các phần chức năng quan trọng
* **Trang 4**

khác nhau cho hạt nhân, cho phép nó cấp phát nhiều bộ nhớ hơn và thực thi các quy trình.

**# /TIP: USE PROTECTED CONTROL TRANSFER**

* Phần cứng hỗ trợ HĐH bằng cách cung cấp các chế độ thực thi khác nhau.Ở chế độ người dùng, các ứng dụng không có toàn quyền truy cập vào tài nguyên phần cứng.Trong chế độ hạt nhân, hệ điều hành có quyền truy cập vào toàn bộ tài nguyên của máy.Hướng dẫn đặc biệt để bẫy vào hạt nhân và trả về từ bẫy trở lạicác chương trình chế độ người dùng cũng được cung cấp, cũng như các hướng dẫn cho phéphệ điều hành để cho phần cứng biết vị trí của bảng bẫy trong bộ nhớ. /
* Hầu hết các hệ điều hành cung cấp khoảng hàng trăm cuộc gọi. Các hệ thống Unix ban đầu không thể hiện một tập hợp con các cuộc gọi chi tiết hơn.
* Để thực hiện một lệnh gọi hệ thống, chương trình phải thực hiện một lệnh bẫy. Thủ tục này khởi chạy mức đặc quyền của hệ thống tới chế độ hạt nhân và khi vào chế độ này, hệ thống giờ đây có thể thực hiện bất kỳ hoạt động đặc quyền nào mà nó cần.
* Khi hoàn tất, HĐH sẽ gửi một lệnh return-from-trap trả về chương trình gọi cho người dùng được chỉ định. Nó trả lại mức đặc quyền trở lại chế độ người dùng.
* Trên x86, bộ xử lý sẽ thực thi bộ đếm chương trình và cờ, và các thanh ghi khác trên ngăn xếp hạt nhân mỗi quá trình. Sau đó, return-fromtrap sẽ bật các giá trị này ra khỏi ngăn xếp.Kernel phải kiểm soát mã nào thực thi trên một bẫy. Điều này là do, để cho phép các chương trình chạy trong hạt nhân, mã phải cụ thể.
* Kernel cũng thiết lập một bảng bẫy khi bắt đầu mỗi lần chạy hệ thống. Bảng này thiết lập các quy tắc cấu hình của phần
* **Trang 5**

cứng cho thời điểm xảy ra các sự kiện nhất định.Hệ điều hành sẽ gửi một lệnh đặc biệt đến phần cứng của trình xử lý bẫy. Điều này hướng dẫn phần cứng lưu trữ vị trí của các trình xử lý cho đến lần khởi động lại tiếp theo.

OS @ boot Hardware

(kernel mode)

initialize trap table

remember address of...

syscall handler

OS @ run Hardware Program

(kernel mode) (user mode)

Create entry for process list

Allocate memory for program

Load program into memory

Setup user stack with argv

Fill kernel stack with reg/PC

return-from-trap

restore regs

(from kernel stack)

move to user mode

jump to main

Run main()...

Call system call

trap into OS

save regs

(to kernel stack)

move to kernel mode

jump to trap handler

Handle trap

Do work of syscall

return-from-trap

restore regs

(from kernel stack)

move to user mode

jump to PC after trap

...

return from main

trap (via exit())

Free memory of process

Remove from process list

*Figure 6.2: Limited Direct Execution Protocol*

* **Trang 6**

#**/TIP: BE WARY OF USER INPUTS IN SECURE SYSTEMS(MẸO: ĐƯỢC TIỀN ĐẦU VÀO CỦA NGƯỜI DÙNG TRONG CÁC HỆ THỐNG BẢO MẬT)**

* Mặc dù chúng tôi đã triển khai nhiều tính năng bảo mật để bảo vệ hệ điều hành, vẫn còn nhiều bước khác cần được thực hiện để triển khai một hệ điều hành an toàn. Một trong số đó là xử lý các đối số tại biên giới cuộc gọi hệ thống.
* Khi gọi một hàm hệ thống, người dùng chỉ định địa chỉ của bộ đệm làm nguồn của lệnh gọi. Hệ điều hành sau đó phải phát hiện và từ chối cuộc gọi nếu nó nhận được một địa chỉ xấu.Để ngăn chặn truy cập trái phép vào đầu vào của người dùng, một hệ thống an toàn phải xử lý chúng một cách hết sức nghi ngờ. Làm như vậy sẽ khiến người dùng có nguy cơ bị tấn công./
* Mỗi cuộc gọi hệ thống được chỉ định một số cuộc gọi hệ thống. Mã người dùng có trách nhiệm đặt số này vào sổ đăng ký hoặc ở một vị trí xác định. Hệ điều hành sẽ kiểm tra số này mỗi khi nó được thực thi và nếu nó hợp lệ, sẽ thực thi mã thích hợp.
* Nếu bạn cố gắng thực thi điều này ở chế độ người dùng, phần cứng có thể sẽ không cho phép bạn. Nếu bạn không biết phải làm gì tiếp theo, hãy suy nghĩ về những điều khủng khiếp có thể xảy ra với hệ thống nếu bạn cố gắng cài đặt một bảng bẫy.
* Bước thứ hai trong quá trình chạy là thiết lập một số thứ trước khi nó thực thi. Điều này cho phép CPU bắt đầu chạy ở chế độ người dùng. Khi tiến trình quyết định thực hiện một lệnh gọi hệ thống, Hệ điều hành sẽ xử lý nó và trả lại quyền kiểm soát cho tiến trình.
* **Trang 7**

**6.3 Problem #2: Switching Between Processes(Vấn đề # 2: Chuyển đổi giữa các quá trình)**

* Vấn đề tiếp theo với thực thi trực tiếp là cố gắng chuyển đổi giữa các quy trình. Nói chung, nó không phải là vấn đề lớn nếu hệ điều hành không chạy, nhưng nếu có, thì nó không chạy. Đây là một vấn đề thực sự và nó chỉ có thể được giải quyết bằng cách yêu cầu hệ điều hành dừng quá trình.

**THE CRUX: LÀM THẾ NÀO ĐỂ TÁI CHẾ ĐỘ KIỂM SOÁT CỦA CPU**

* Làm thế nào hệ điều hành có thể lấy lại quyền kiểm soát CPU để nó có thểchuyển đổi giữa các quy trình?

**A Cooperative Approach: Wait For System Calls(Phương pháp tiếp cận hợp tác: Chờ cuộc gọi hệ thống)**

* Cách tiếp cận hợp tác là cách tiếp cận hệ điều hành kiểu Unix. Nó liên quan đến việc hệ điều hành từ bỏ bộ xử lý khi quá lâu và cho phép hệ thống thực hiện các tác vụ khác.
* Về cơ bản, hầu hết các quy trình thực hiện các cuộc gọi hệ thống tới HĐH để kiểm soát CPU. Điều này có nghĩa là họ có thể chạy các quy trình khác mà không cần chuyển giao quyền kiểm soát một cách rõ ràng đối với Hệ điều hành.
* Khi một ứng dụng cố gắng truy cập vào tài nguyên hệ thống mà nó không thể truy cập được, hệ điều hành sẽ kết thúc quá trình và trả lại quyền kiểm soát cho PC.
* **Trang 8**

**A Non-Cooperative Approach: The OS Takes Control(Phương pháp tiếp cận không hợp tác: Hệ điều hành giành quyền kiểm soát)**

/ **CRUX: CÁCH KIỂM SOÁT MÀ KHÔNG CẦN HỢP TÁC**

* Làm thế nào hệ điều hành có thể giành quyền kiểm soát CPU ngay cả khi các quy trình không đượchợp tác xã? Hệ điều hành có thể làm gì để đảm bảo quá trình giả mạo không diễn raqua máy? /
* Nếu không có sự trợ giúp thêm từ phần cứng, hệ điều hành không thể làm được gì nhiều khi một quy trình từ chối thực hiện cuộc gọi đến hệ thống. Trong hầu hết các trường hợp, giải pháp duy nhất là khởi động lại máy.
* Câu trả lời rất đơn giản: Ngắt bộ định thời là một thủ tục lập trình có thể được lập trình để nâng cao một loạt thời gian ngắt, có thể được thực hiện sau mỗi mili giây. Khi bộ đếm thời gian ngắt, quá trình hiện tại bị dừng và hệ điều hành sau đó có thể kiểm soát nó.Hệ điều hành sau đó có thể cảm thấy an toàn khi quyền kiểm soát của nó cuối cùng sẽ được trả lại.
* Điều này cũng đúng nếu bộ hẹn giờ bị tắt (đây cũng là một hoạt động đặc quyền). Khi phần cứng nhận thấy rằng một ngắt đã xảy ra, nó sẽ tự động dừng chương trình đang chạy và lưu trạng thái của nó.

**#/ TIP: DEALING WITH APPLICATION MISBEHAVIOR**

* Hệ điều hành thường phải đối phó với các quy trình hoạt động sai, nhữngmà thông qua thiết kế (độc hại) hoặc tai nạn (lỗi) cố gắng làm điều gì đó mà họ không nên làm. Trong các hệ thống hiện đại, cách hệ điều hành cố gắng xử lý hành vi vi phạm như vậy chỉ đơn giản là chấm dứt người vi phạm. Một tấn công và bạn đã ra ngoài! Có lẽ tàn bạo, nhưng hệ điều hành nên làm gì khác khi bạn cố gắng truy cập bộ nhớ bất hợp pháp hoặc thực hiện một lệnh bất hợp pháp? /
* **Trang 9**

**Lưu và khôi phục ngữ cảnh**

Nếu quyết định chuyển đổi được thực hiện, thì hệ điều hành sẽ thực thi mức thấp

đoạn mã mà chúng tôi gọi là công tắc ngữ cảnh. Một công tắc ngữ cảnh là

đơn giản về mặt khái niệm: tất cả những gì hệ điều hành phải làm là lưu một vài giá trị thanh ghi

cho quá trình hiện đang thực thi (ví dụ: trên ngăn xếp hạt nhân của nó)

và khôi phục một số ít cho quá trình sắp thực thi (từ hạt nhân của nó

cây rơm). Bằng cách đó, hệ điều hành sẽ đảm bảo rằng khi trả về từ bẫy

hướng dẫn cuối cùng được thực thi, thay vì quay trở lại quá trình đã

đang chạy, hệ thống tiếp tục thực hiện một quy trình khác.

Để lưu bối cảnh của quá trình hiện đang chạy, HĐH sẽ xuất hiện một số mã lắp ráp cấp thấp để lưu các mục đích chung regis ters, PC và con trỏ ngăn xếp hạt nhân của quá trình hiện đang chạy,

và sau đó khôi phục các đăng ký, PC và chuyển sang ngăn xếp hạt nhân cho

quy trình sắp được thực thi. Bằng cách chuyển đổi các ngăn xếp, hạt nhân đi vào

gọi đến mã chuyển mạch trong ngữ cảnh của một quy trình (quy trình đã bị gián đoạn) và trả về trong ngữ cảnh của quy trình khác (quy trình sắp được thực thi

một). Khi hệ điều hành cuối cùng thực hiện lệnh trả về từ bẫy,

**MẸO: SỬ DỤNG TIMER INTERRUPT ĐỂ KIỂM SOÁT TÁI TẠO**

Việc bổ sung ngắt hẹn giờ cho phép HĐH có thể chạy lại

trên CPU ngay cả khi các quy trình hoạt động theo kiểu bất hợp tác. Vì vậy, điều này

tính năng phần cứng là điều cần thiết trong việc giúp hệ điều hành duy trì quyền kiểm soát

cỗ máy.

* **Trang 10**

**MẸO: REBOOT LÀ HỮU ÍCH**

Trước đó, chúng tôi đã lưu ý rằng giải pháp duy nhất cho vòng lặp vô hạn (và tương tự

hành vi) được ưu tiên hợp tác là khởi động lại máy. Trong khi

bạn có thể chế giễu vụ hack này, các nhà nghiên cứu đã chỉ ra rằng khởi động lại (hoặc trong gen eral, bắt đầu lại một số phần mềm) có thể là một công cụ cực kỳ hữu ích trong

xây dựng hệ thống mạnh mẽ [C + 04].

Cụ thể, khởi động lại rất hữu ích vì nó chuyển phần mềm trở lại trạng thái đã biết

và trạng thái có thể được thử nghiệm nhiều hơn. Khởi động lại cũng khôi phục lại các nguồn cũ bị rò rỉ hoặc cũ (ví dụ: bộ nhớ) mà có thể khó xử lý. Cuối cùng,

khởi động lại dễ dàng tự động hóa. Vì tất cả những lý do này, không có gì lạ

trong các dịch vụ Internet cụm quy mô lớn cho phần mềm quản lý hệ thống

khởi động lại định kỳ các bộ máy để đặt lại chúng và do đó

có được những lợi thế được liệt kê ở trên.

Vì vậy, lần sau khi bạn khởi động lại, bạn không chỉ thực hiện một số vụ hack xấu xí.

Thay vào đó, bạn đang sử dụng phương pháp tiếp cận đã được kiểm tra thời gian để cải thiện hành vi

của một hệ thống máy tính. Làm tốt!

* **Trang 11**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **OS @ boot Phần cứng**  **(chế độ hạt nhân)** | Phần cứng |  | | **khởi tạo bảng bẫy** | ghi nhớ địa chỉ của ...  trình xử lý cuộc gọi syscall  bộ xử lý hẹn giờ |  | | **bắt đầu hẹn giờ ngắt** | bắt đầu hẹn giờ  ngắt CPU trong X giây |  | | **OS @ boot Phần cứng**  **(chế độ hạt nhân)** | **Phần cứng** | **Chương trình (chế độ người dùng)** | |  |  | **Quy trình A** | | Xử lý bẫy  Chuyển đổi cuộc gọi () quy trình  lưu regs (A) → proc t (A)  khôi phục regs (B) ← proc t (B)  chuyển sang k-stack (B)  **return-from-trap (vào B)** |  | **hẹn giờ ngắt**  lưu regs (A) → k-stack (A)  chuyển sang chế độ hạt nhân  nhảy vào bẫy |  | |  | khôi phục regs (B) ← k-stack (B) chuyển sang  chế độ người dùng chuyển sang PC của B |  | |  |  | **Quy trình B** | |

Hình 6.3: Giao thức thực thi trực tiếp có giới hạn (Ngắt bộ định thời)

Trang 12

**6.4 Lo lắng về đồng tiền?**

Một điều đơn giản mà hệ điều hành có thể làm là có thể ngắt trong quá trình xử lý ngắt; làm như vậy đảm bảo rằng khi

một ngắt đang được xử lý, không có ngắt nào khác sẽ được chuyển đến CPU.

Tất nhiên, hệ điều hành phải cẩn thận khi làm như vậy; vô hiệu hóa gián đoạn cho

quá lâu có thể dẫn đến mất ngắt, điều này không tốt (về mặt kỹ thuật).

Hệ điều hành cũng đã phát triển một số

khóa các sơ đồ để bảo vệ quyền truy cập đồng thời vào cấu trúc dữ liệu nội bộ.

Điều này cho phép nhiều hoạt động đang diễn ra bên trong hạt nhân tại

đồng thời, đặc biệt hữu ích trên các bộ đa xử lý.

1 # void swtch(struct context \*\*old, struct context \*new);

2 #

3 # Save current register context in old

4 # and then load register context from new.

5 .globl swtch

6 swtch:

7 # Save old registers

8 movl 4(%esp), %eax # put old ptr into eax

9 popl 0(%eax) # save the old IP

10 movl %esp, 4(%eax) # and stack

11 movl %ebx, 8(%eax) # and other registers

12 movl %ecx, 12(%eax)

13 movl %edx, 16(%eax)

14 movl %esi, 20(%eax)

15 movl %edi, 24(%eax)

16 movl %ebp, 28(%eax)

17

18 # Load new registers

19 movl 4(%esp), %eax # put new ptr into eax

20 movl 28(%eax), %ebp # restore other registers

21 movl 24(%eax), %edi

22 movl 20(%eax), %esi

23 movl 16(%eax), %edx

24 movl 12(%eax), %ecx

25 movl 8(%eax), %ebx

26 movl 4(%eax), %esp # stack is switched here

27 pushl 0(%eax) # return addr put in place

28 ret # finally return into new ctxt

* Hình 6.4: Mã chuyển đổi ngữ cảnh xv6
* **Trang 13**

**6.5 Tóm tắt**

**BÊN NGOÀI: CHUYỂN ĐỔI TIẾP THEO BAO LÂU**

Một câu hỏi tự nhiên mà bạn có thể có là: bao lâu một cái gì đó giống như một chuyển đổi ngữ cảnh mất? Hoặc thậm chí là một cuộc gọi hệ thống? Đối với những người trong số các bạn có thói quen điên cuồng, có một công cụ được gọi là lmbench [MS96] để đo lường chính xác những cũng như một số thước đo hiệu suất khác có thể phù hợp. Kết quả đã được cải thiện khá nhiều theo thời gian, bộ xử lý theo dõi đại khái màn biểu diễn. Ví dụ, vào năm 1996, chạy Linux 1.3.37 trên 200 MHz CPU P6, các cuộc gọi hệ thống mất khoảng 4 micro giây và một công tắc ngữ cảnh khoảng 6 micro giây [MS96]. Các hệ thống hiện đại hoạt động tốt hơn gần như một hoặc trăm độ lớn, với kết quả dưới micro giây trên các hệ thống có Bộ xử lý 2 hoặc 3 GHz. Cần lưu ý rằng không phải tất cả các hành động của hệ điều hành đều theo dõi CPU trên mỗi hình thức. Theo quan sát của Ousterhout, nhiều hoạt động của hệ điều hành là bộ nhớ chuyên sâu và băng thông bộ nhớ không được cải thiện đáng kể như tốc độ xử lý theo thời gian [O90]. Do đó, tùy thuộc vào khối lượng công việc của bạn, mua bộ xử lý mới nhất và tốt nhất có thể không tăng tốc hệ điều hành của bạn vì nhiều như bạn có thể hy vọng.

* **Trang 14**

**BÊN TRONG: ĐIỀU KHOẢN VIRTUALIZATION CHÍNH CỦA CPU (CƠ CHẾ)**

**•** CPU phải hỗ trợ ít nhất hai chế độ thực thi: chế độ người dùng được nghiêm ngặt hơn và chế độ hạt nhân đặc quyền (không hạn chế).

**•** Các ứng dụng người dùng điển hình chạy ở chế độ người dùng và sử dụng lệnh gọi hệ thống

để bẫy vào hạt nhân để yêu cầu các dịch vụ hệ điều hành.

**•** Lệnh bẫy lưu cẩn thận trạng thái thanh ghi, thay đổi trạng thái kho cứng thành chế độ hạt nhân và chuyển vào hệ điều hành đến một trạng thái được chỉ định trước

đích: bảng bẫy.

• Khi hệ điều hành hoàn tất phục vụ cuộc gọi hệ thống, nó sẽ trở lại người dùng

chương trình thông qua một lệnh đặc biệt khác từ bẫy, cho phép đặc quyền và trả lại quyền kiểm soát cho lệnh sau bẫy

đã nhảy vào hệ điều hành.

• Các bảng bẫy phải được HĐH thiết lập tại thời điểm khởi động, và

chắc chắn rằng chúng không thể được sửa đổi dễ dàng bởi các chương trình người dùng. Tất cả các

đây là một phần của giao thức thực thi trực tiếp hạn chế chạy

chương trình hiệu quả nhưng không mất quyền kiểm soát hệ điều hành.

• Khi một chương trình đang chạy, hệ điều hành phải sử dụng các cơ chế phần cứng

để đảm bảo chương trình người dùng không chạy mãi mãi, cụ thể là bộ đếm thời gian

ngắt. Cách tiếp cận này là một cách tiếp cận không hợp tác đối với CPU

lập kế hoạch.

• Đôi khi HĐH, trong thời gian ngắt hẹn giờ hoặc cuộc gọi hệ thống, có thể

muốn chuyển từ chạy quy trình hiện tại sang một quy trình khác,