ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

- -

HỆ ĐIỀU HÀNH

BÀI TẬP LỚN SỐ 1

SYSTEM CALL

Giáo viên hướng dẫn: Nguyễn Minh Trí
Sinh viên: 160852 - Huỳnh Sâm Hà

Ho Chi Minh City, 3/2018

Mục lục

1 Bước 1: Prepare Linux Kenel

1.1 Bước 1.1: Preparation

- Ở bước này, ta thiết lập máy ảo (ở đây sử dụng Virtual Box trên hệ điều hành Window 10) và tải file ảnh Ubuntu tại đường dẫn trong assignment.
- Sau đó cài các các gói cần thiết khi mới sử dụng ubuntu bằng lệnh install gói build-essential.
- Cài đặt gói kernel-package.
- Tạo thư mục /kernelbuild cho việc chứa kernel.

QUESTION: Why we need to install kernel-package?

kerel-package là một gói tiện ích cho việc xây dựng 1 kernel trong linux. Gói này sẽ tự động hóa các bước thông thường được yêu cầu để biên dịch và cài đặt 1 kernel tùy chỉnh.

- Tải kernel source và giải nén vào thư mục /kernelbuild.
- Cài package openssl và libssl-dev

QUESTION: Why we have to use another kernel source from the server such as http://www.kernel.org, can we compile the original kernel (the local kernel on the running OS) directly?

==========

1.2 Bước 1.2: Configuration

- Làm như các bước trong assignment.
- Sau khi thêm mssv (1610852) trong lúc cấu hình trên GUI, ta kiểm tra lại trong file .config xem có dòng CONFIG_LOCALVERSION=".1610852" chưa để kiểm tra thành công của bước này.

2 Bước 2: System Call - procsched

- Ở bước này, ta xây dựng kernel thông qua 2 bước chính là kiểm tra thử với thư viện kernel module, sau đó hiện thực và cấu hình cần thiết trước khi biên dịch kernel mới.
- Đầu tiên ta tìm hiểu các định nghĩa struct có sẵn trong các thư viện như assignment đã viết.

2.1 Bước 2.1: Kernel Module

- Ở bước này, ta sử dụng thư viện kernel module để test thử chương trình của ta chạy đúng không.
- Làm như các bước trong tut ở đường dẫn, thử với chương trình in ra 'Hello World':
  + Copy Makefile như tut.
  + Viết chương trình in ra dòng chữ 'Hello World'.
  + Gọi make để tạo file .ko
  + Gọi lệnh sudo insmod ./hello-1.ko để insert kernel này vào.
  + Kiểm tra trong xem đã có kernel này chưa bằng lệnh cat /proc/modules | grep hello, thấy output có hello-1.
  + Xóa bỏ kernel này bằng lệnh sudo rmmod hello-1. Kiểm tra lại bằng lệnh cat ở trên thấy không có output, tức là đã xóa bỏ kernel này.
  + Xem trong log hệ thống có dòng chữ 'Hello World' không, bằng lệnh dmesg, thấy có tức là kernel có hoạt động.
- Sau khi test thử với chương trình đơn giản trên, ta hiện thực chương trình cần làm.
  + Viết file như assignment.
  + Tìm các method để giải quyết yêu cầu. Mã nguồn nưu file đính kèm trong báo cáo.
  + Kiểm tra lại thấy có in ra kết quả.

2.2 Bước 2.2: Prototype

- Đọc định nghĩa prototype để biết cách hoạt động của hàm.
- Hàm procsched trả về 0 nếu tìm thấy và trả về thông tin trong biến con trỏ info. Ngược lại trả về 1.
- Sửa lại source code ở bước kiểm trả kernel ở trên theo đúng định nghĩa của hàm.
- Kiểm tra lại kết quả với các pid ngẫu nhiên.

2.3 Bước 2.3: Implementation

- Cấu hình như assignment.
- Hiện thực code như ở bước kiểm tra kernel.

QUESTION: What is the meaning of other parts, i.e. i386, procsched, and sys procsched?

Mỗi system call đươc định nghĩa trong 1 hàng với các đối số theo thứ tự number, ABI, name, entry point và compat entry point. Trong đó mỗi đối số mang ý nghĩa:

- <**number**>: Đây là số định danh riêng cho từng syscall ở mỗi loại máy 32-bit hoặc 64-bit
- <**abi**>: Application Binary Interface, mang giá trị i386 cho hệ máy 32-bit và ========== cho hệ máy 64-bit
- <**name**>: ==========
- <**entry point**>: ==========
- <**compat entry point**>: ==========

QUESTION: What is the meaning of each line added in file include/linux/syscalls.h?

- struct proc_segs;

  Dòng này khai báo 1 struct trong file header một cách thông thường.

- asmlinkage long sys_procsched(int pid, struct proc_segs * info);

  + Dòng này khai báo 1 hàm trong file header, nhưng ở đây là 1 hàm asmlinkage. Từ khóa asmlinkage nói cho trình biên dịch biết tìm các đối số hàm trên stack của CPU thay vì trên các thanh ghi.

  + Tại sao lại phải làm như vậy? System calls là các dịch vụ mà userspace có thể gọi request tới kernel để thực hiện một số công việc, và các công việc này nằm trong kernel space. Các hàm xử lý các công việc này không giống như các hàm thông thường, các đối số được truyền thông qua program stack, chứ không phải qua thanh ghi như các hàm thông thường khác.

3 Bước 3: Compiling Linux Kernel

3.1 Bước 3.1: Build the configured kernel

Ở bước này ta thực hiện lệnh make và make modules như assignment để compile và build kernel.

QUESTION: What is the Omeaning of these two stages, namely "make" and "make modules"?

- make:

  + Dùng để biên dịch kernel source mà ta đã tải ở các bước trên. Nhưng ở đây kernel này ta đã thay đổi. Sau bước này sẽ tạo ra file vmlinuz.

  + *vmlinuz*: File vmlinuz là 1 file thực thi kernel linux. File này đã được nén lại và có thể giải nén và load vào hệ điều hành trong vùng nhớ.

- make modules:

  Dòng này dùng để biên dịch kernel modules.

3.2 Bước 3.2: Installing the new kernel

Làm như các bước trong assignment.

3.3 Bước 3.3: Testing

Làm như các bước trong assignment.

QUESTION:Why this program could indicate whether our system works or not?

Ta gọi system call thông qua thư viện <sys/syscall.h> bằng lệnh syscall ([number_32], [pid], info); với number_32 là số định danh của procsched system call mà ta đã khai báo (ở đây là 377) trong file syscall 32.tlb.
Nếu kernel ta build thất bại, chương trình sẽ không hoạt động. Ngược lại, syscall sẽ trả về giá trị trong con trỏ info được truyền vào. Do đó với chương trình này ta có thể kiểm tra xem kernel đã được build thành công không.

4 Bước 4: Wrapper

Ở bước này ta thực hiện gói lại thành thư viện tiện cho người dùng cuối.

QUESTION:Why we have to re-define proc segs struct while we have already defined it inside the kernel?

Struct proc_segs mà ta đã định nghĩa chỉ được sử dụng trong kerlen space (nằm trong kernel). Do đó để sử dụng bên ngoài userspace, ta cần định nghĩa lại một struct tương đương như một module tiện ích cho người dùng tiện sử dụng.

QUESTION: Why root privilege (e.g. adding sudo before the cp command) is required to copy the header file to /usr/include?

Lệnh sudo (superuserd do) được sử dụng như quyền của user root. Còn thư mục /usr/ được sử dụng bởi user root, do đó để thực hiện ghi chép file vào thư mục này ta cần dùng quyền của root bằng cách dùng sudo

QUESTION:Why we must put -share and -fpic option into gcc command?

Ý nghĩa của mỗi lệnh gcc là:

- -shared: dùng để tạo ra một thư viện shared (shared library).
- -fpic: dùng để tạo ra PIC (position-independent code) tương thích cho việc sử dụng một shared library.

Ta đang muốn tạo ra một file Shre Object (file .so, cụ thể libprocsched.so), file này có thể được liên kết tới các chương trình khác sử dụng thư viện của ta. Do đó ta phải thêm 2 option này như mục đích của chúng.

Tài liệu

[Network] kernel-package: http://man.he.net/man5/kernel-package

[Network] https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_C_libraries.html