

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH



HỆ ĐIỀU HÀNH

GVHD: Nguyễn Hoài Nam
SV: Triệu Tấn Hùng - 1812475

TP. HỒ CHÍ MINH, THÁNG 05/2020



Mục lục

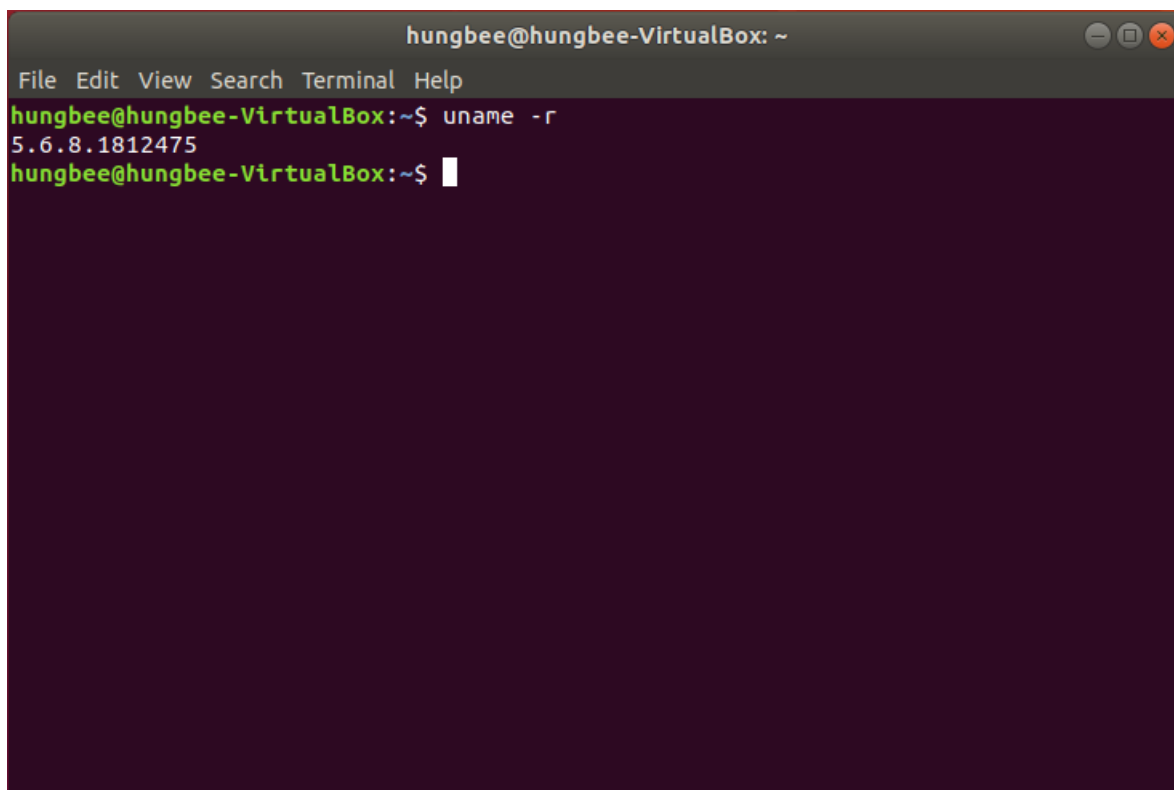
1	Quy trình thực hiện	2
1.1	Compiling Linux Kernel	2
1.2	System call	3
1.3	Test	3
1.4	Validation	4
2	Trả lời Question	4
3	Soure code	7
4	Tài liệu tham khảo	7

1 Quy trình thực hiện

1.1 Compiling Linux Kernel

- Install the core packages:
 - `sudo apt-get update`
 - `sudo apt-get install build-essential`
 - `sudo apt-get install kernel-package`
- Create a kernel compilation directory:
 - `mkdir ~/kernelbuild`
 - `cd ~/kernelbuild`
- Download the kernel source:
 - `wget https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/linux-5.6.8.tar.gz`
 - `tar -zxvf linux-5.6.8.tar.gz`
- Configuration:
 - `cp /boot/config-$(uname -r) ~/kernelbuild/linux-5.6.8/.config`
 - `sudo apt-get install fakeroot ncurses-dev xz-utils bc flex libelf-dev bison`
 - `cd linux-5.6.8`
 - `make nconfig`
 - General Setup
- Build the configured kernel:
 - `make -j 4`
 - `make -j 4 modules`
 - `sudo make -j 4 modules_install`
 - `sudo make -j 4 install`
 - `sudo reboot`

Kết quả:



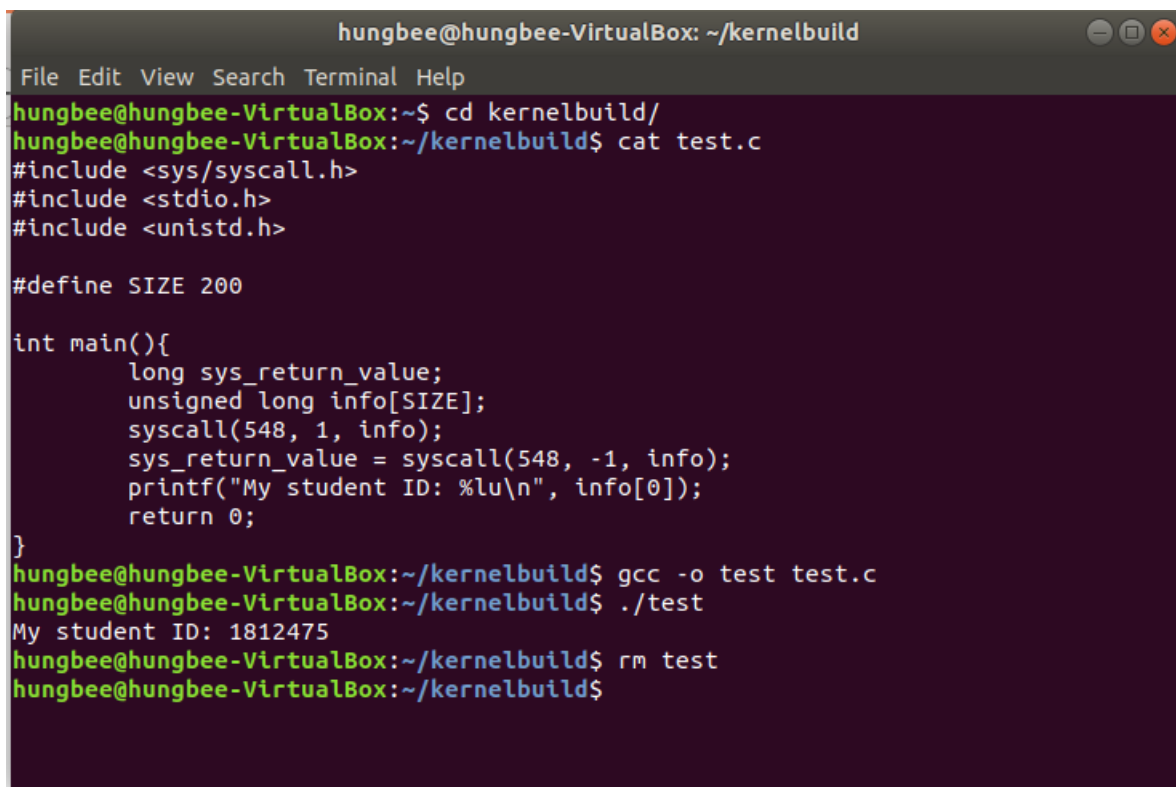
```
hungbee@hungbee-VirtualBox: ~  
File Edit View Search Terminal Help  
hungbee@hungbee-VirtualBox:~$ uname -r  
5.6.8.1812475  
hungbee@hungbee-VirtualBox:~$
```

Hình 1: Build Kernel

1.2 System call

```
+ cd ~/kernelbuild/linux-5.6.8  
+ mkdir get_proc_info  
+ cd get_proc_info  
+ touch sys_get_proc_info.c  
+ touch Makefile  
+ echo "obj-y := sys_get_proc_info.o"» Makefile  
+ Add new system call to the system call header file  
+ make -j 8  
+ make modules -j 8  
+ sudo make modules_install  
+ sudo make install  
+ sudo reboot
```

1.3 Test



```
hungbee@hungbee-VirtualBox: ~/kernelbuild
File Edit View Search Terminal Help
hungbee@hungbee-VirtualBox:~$ cd kernelbuild/
hungbee@hungbee-VirtualBox:~/kernelbuild$ cat test.c
#include <sys/syscall.h>
#include <stdio.h>
#include <unistd.h>

#define SIZE 200

int main(){
    long sys_return_value;
    unsigned long info[SIZE];
    syscall(548, 1, info);
    sys_return_value = syscall(548, -1, info);
    printf("My student ID: %lu\n", info[0]);
    return 0;
}
hungbee@hungbee-VirtualBox:~/kernelbuild$ gcc -o test test.c
hungbee@hungbee-VirtualBox:~/kernelbuild$ ./test
My student ID: 1812475
hungbee@hungbee-VirtualBox:~/kernelbuild$ rm test
hungbee@hungbee-VirtualBox:~/kernelbuild$
```

Hình 2: Test

1.4 Validation

2 Trả lời Question

- Why we need to install kernel-package?

Answer:

Bởi vì tại đây chúng ta có rất nhiều phiên bản hạt nhân (kernel) để cho chúng ta lựa chọn sao cho phù hợp với cấu hình phần cứng của máy ảo hay máy thật của bạn

- 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?

Answer:

Được, bạn có thể biên dịch hạt nhân gốc (the original kernel) trong tệp của OS đang sử dụng, vì hạt nhân mặc định được vận chuyển với Debian xử lý hầu hết các cấu hình. Ngoài ra, Debian thường cung cấp một số hạt nhân thay thế, bạn cần kiểm tra trước hạt nhân này tương thích tốt với cấu hình phần cứng của bạn, tuy nhiên những lợi ích cụ thể biên dịch từ hạt nhân (kernel) mới từ server là để:

- Xử lý các nhu cầu phần cứng đặc biệt, hoặc xung đột phần cứng với hạt nhân được cung cấp trước

```
hungbee@hungbee-VirtualBox: ~/kernelbuild/Wrapper
File Edit View Search Terminal Help
hungbee@hungbee-VirtualBox:~$ cd kernelbuild/Wrapper/
hungbee@hungbee-VirtualBox:~/kernelbuild/Wrapper$ sudo cp ./get_proc_info.h /usr
/include
[sudo] password for hungbee:
hungbee@hungbee-VirtualBox:~/kernelbuild/Wrapper$ gcc -shared -fpic get_proc_inf
o.c -o libget_proc_info.so
hungbee@hungbee-VirtualBox:~/kernelbuild/Wrapper$ sudo cp ./libget_proc_info.so
/usr/lib
hungbee@hungbee-VirtualBox:~/kernelbuild/Wrapper$ gcc validation.c -lget_proc_i
nfo -o main
hungbee@hungbee-VirtualBox:~/kernelbuild/Wrapper$ ./main
PID: 2032
studentID: 1812475
proc.pid: 2032
proc.name: main
parent_proc.pid: 1975
parent_proc.name: bash
oldest_child_proc.pid: 0
oldest_child_proc.name:
hungbee@hungbee-VirtualBox:~/kernelbuild/Wrapper$
```

Hình 3: Validation

- Sử dụng các tùy chọn sử dụng hạt nhân mà không được hỗ trợ trong các hạt nhân được cung cấp trước (chẳng hạn như hỗ trợ bộ nhớ cao)
- Tối ưu hóa hạt nhân bằng cách loại bỏ các trình điều khiển vô ích để tăng tốc độ khởi động
- Tạo ra một monolithic thay vì một hạt nhân đã được modularized
- Chạy một cập nhật hạt nhân hoặc dành cho nhà phát triển

- What is the meaning of these two stages, namely “make” and “make modules”? What are created and what for?

Answer:

Installing the new kernel

```
$ sudo make modules_install
```

or

```
$ sudo make -j 4 modules_install
```

Then install the kernel itself:

```
$ sudo make install
```

or

```
$ sudo make -j 4 install
```

```
$ sudo reboot  
done
```

- What is the meaning of fields of the line that just add to the system call table (548, 64, get_proc_info, i.e.)

Answer:

-584: [Number] Tất cả các syscalls được xác định bởi một số duy nhất. Để gọi một syscall, chúng ta nói với kernel để gọi theo số chứ không phải bằng tên của nó.

-64: [ABI] Application Binary Interface - là interface giữa hai chương trình modules, một trong số đó thường là thư viện hoặc điều hành, ở mức mã máy phổ biến x32, x64, i386.

-get_proc_info: [name]: đây là tên của syscall.

-sys_get_proc_info: [entry point]: Điểm truy cập, là tên hàm để gọi để xử lý syscall. Quy ước đặt tên hàm là tên của syscall có tiền tố _syscall.

- What is the meaning of each line above?

```
struct proc_info;  
struct procinfo;  
asm linkage long sys_get_proc_info(pid_t pid, struct procinfo * info);
```

Answer:

-Hàm xử lý syscall trả về long

-Asmlinkage là một tag được định nghĩa với một số gcc biên dịch cho trình biên dịch biết rằng hàm không mong chờ tìm thấy tất cả đối tượng trong thanh ghi một cách tối ưu phổ biến. Những chỉ trên stack của CPU. Syscall tiếp nhận đối số number đầu tiên, cho phép 4 đối số được truyền vào hệ thống thực, các đối số này nằm trên stack. Tất cả syscall đánh dấu bởi asmlinkage tag nên chúng nhìn vào stack để tìm đối số. Nó cũng được sử dụng cho phép gọi một hàm từ assembly files.

- Why we have to redefine procinfo and proc_info struct while we have already defined it inside the kernel?

Answer:

```
struct proc_info{  
    pid_t pid;  
    char name[16];  
};
```

Thông tin quá trình được lưu trữ trong proc_info chứa:

-pid: Pid của quá trình

-name: Tên của chương trình được thực hiện

```
struct procinfo{  
    long studentID;  
    struct proc_info proc;  
    struct proc_info parent_proc;  
    struct proc_info oldest_child_proc;  
};
```

procinfo chứa thông tin về ba quy trình:

-proc: Quy trình hiện tại hoặc quy trình với PID

-parent_proc: Cha mẹ của quá trình đầu tiên

-oldest_child_proc, Quá trình con sớm nhất của quá trình đầu tiên

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

Answer:

Việc copy file vào trong /usr/include để thiết lập một môi trường làm việc mới, tất cả những gì bạn phải làm là lấy từ kho lưu trữ. Thông thường mọi người đưa những thông tin đó vào Makefile. Bằng cách này, bạn có thể đặt mã nguồn của các thư viện bên ngoài vào kho lưu trữ của mình và chỉ cần yêu cầu trình biên dịch cũng tìm kiếm các tiêu đề ở đó.

- Why we must put -shared and -fpic option into gcc command?

Answer:

-Tùy chọn -shared nhằm tạo ra tệp tin cho thư viện chia sẻ. Thư mục chia sẻ là tập hợp các tệp đối tượng. Các tệp đối tượng không được kết hợp với các chương trình tại thời điểm biên dịch, đồng thời, chúng không được sao chép vĩnh viễn vào tệp thực thi cuối cùng.

-Tùy chọn -fpic cho phép tạo "position independent code" một yêu cầu cho các thư viện dùng chung. Sử dụng -fpic để tạo mã.

3 Soure code

Các phần của code đã có comment để tiện cho việc tìm hiểu. Bạn có thể vào file soure code để xem thêm.

4 Tài liệu tham khảo

- <https://medium.com/@ssreehari/implementing-a-system-call-in-linux-kernel-4-7-1-6f98250a8c38>
- <https://shanetully.com/2014/04/adding-a-syscall-to-linux-3-14/>
- [edium.com/anubhav-shrimal/adding-a-hello-world-system-call-to-linux-kernel-dad32875872](https://medium.com/anubhav-shrimal/adding-a-hello-world-system-call-to-linux-kernel-dad32875872)
- <http://gitref.org/>
- <https://cs-fundamentals.com/c-programming/static-and-dynamic-linking-in-ccreate-and%20use-shared-libraries>


```
C validation.c C sys_get_proc_info.c X
home > hungbee > kernelbuild > linux-5.6.8 > get_proc_info > C sys_get_proc_info.c > ...
15
16 SYSCALL_DEFINE2(get_proc_info, pid_t, pid, struct procinfo *, info)
17 {
18     info->studentID = 1812475;
19
20     //Kiểm tra pid có hợp lệ hay không.
21     if (pid == -1) {
22         pid = current->pid;
23     }
24     struct task_struct *proce;
25     for_each_process (proce) {
26         if (proce->pid == pid) {
27             struct task_struct *cProce;
28             cProce = list_first_entry_or_null(&proce->children, struct task_struct, sibling); // Tao ra oldest_ch
29
30             //Truy xuất thông tin của proc
31             info->proc.pid = proce->pid;
32             strcpy(info->proc.name, proce->comm);
33
34             //Truy xuất thông tin của parent_proc
35             if (proce->real_parent == NULL) {
36                 info->parent_proc.pid = 0;
37                 strcpy(info->parent_proc.name, ""); //info->parent_proc.name = "";
38             }
39             else {
40                 info->parent_proc.pid = proce->real_parent->pid;
41                 strcpy(info->parent_proc.name, proce->real_parent->comm); //info->parent_proc.name = proce->real_
42             }
43
44             //Truy xuất thông tin của oldest_child_proc
45             if (cProce == NULL) {
46                 info->oldest_child_proc.pid = 0;
47                 strcpy(info->oldest_child_proc.name, ""); //info->oldest_child_proc.name = "";
48             }
49             else {
50                 info->oldest_child_proc.pid = cProce->pid;
51                 strcpy(info->oldest_child_proc.name, cProce->comm); //info->oldest_child_proc.name = cProce->comm
52             }
53         }
54     }
55 }
56
```

Hình 4: sys_get_proc_info code

```
C validation.c C sys_get_proc_info.c C get_proc_info.c X
C get_proc_info.c > get_proc_info(pid_t, procinfo *)
1 #include "get_proc_info.h"
2 #include <linux/kernel.h>
3 #include <sys/syscall.h>
4 #include <unistd.h>
5
6 long get_proc_info(pid_t pid, struct procinfo * info) {
7     long sysvalue;
8     sysvalue = syscall(548, pid, info);
9     return sysvalue;
10 }
```

Hình 5: get_proc_info code

```
C validation.c X
C validation.c > main()
1
2 #include <get_proc_info.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <stdio.h>
6 #include <stdint.h>
7
8 int main(){
9     pid_t mypid = getpid();
10    printf("PID: %d\n", mypid);
11    struct procinfo info;
12
13    if(get_proc_info(mypid, &info) == 0){
14        printf("studentID: %ld\n", info.studentID); //xuat Student id:
15        printf("proc.pid: %d\n", info.proc.pid); //xuat pid của proc
16        printf("proc.name: %s\n", info.proc.name); //xuat name của proc
17        printf("parent_proc.pid: %d\n", info.parent_proc.pid); //xuat pid của parent_proc
18        printf("parent_proc.name: %s\n", info.parent_proc.name); //xuat name của parent_proc
19        printf("oldest_child_proc.pid: %d\n", info.oldest_child_proc.pid); //xuat pid của oldest_child_proc
20        printf("oldest_child_proc.name: %s\n", info.oldest_child_proc.name); //xuat name của oldest_child_proc
21    } else {
22        printf("Cannot get information from the process %d\n", mypid);
23    }
24 }
```

Hình 6: validation code