Đại học Quốc gia Thành phố Hồ Chí Minh
Đại học Khoa Học Tự Nhiên
Khoa Công nghệ thông tin

[HỆ ĐIỀU HÀNH]

TÌM HIỂU VÀ LẬP TRÌNH LINUX KERNEL MODULE

[GIÁO VIÊN HƯỚNG DẪN]
Lê Viết Long
Phạm Tuấn Sơn
[SINH VIÊN]
Trần Xuân Sơn – 19127321

I. Tổng quan về đồ án

STT	Tên thành phần	Hoàn thành
1	Viết linux kernel module tạo số ngẫu nhiên	100%

Thành phần của bài làm gồm:

- Random.c: Source code của kernel module tạo số ngẫu nhiên
- testRandom.c: Source code chương trình test module Random.
- README.md: File README dạng markdown. README được hiển thị ở đây (https://github.com/txuanson/linux_rand_module#readme).
- Report: File báo cáo của đồ án.

II. Chi tiết:

1. Cấu hình biên dịch:

- Để tiện cho quá trình biên dịch, thay vì phải viết những câu lệnh dài thì ta tạo ra các script để biên dịch:
    - Makefile:
        - Để chương trình có thể thực thi trong linux, ta cần biên dịch chương trình và Makefile là 1 dạng script dùng để biên dịch chương trình.
    - Kbuild:
        - Kbuild là chương trình được dùng bởi Linux kernel để build các module và chọn các flag gcc để biên dịch.

2. Khởi tạo module:

- Gồm 4 bước, với mỗi bước nếu gặp lỗi thì sẽ phải hủy các thay đổi (tương tự với việc hủy module) đã tạo ra (theo thứ tự ngược lại) cho kernel ở các bước trước đó và trả lại mã lỗi là số âm (ở đây ta dùng -1 làm mặc định).
    - Đăng ký số hiệu file thiết bị:
        - Sử dụng hàm

```
int alloc_chrdev_region(dev_t *first, unsigned int firstminor,
unsigned int cnt, char *name);
```

- Hàm này giúp đăng ký một cặp số <***major, minor***> là số hiệu cho file thiết bị và được lưu vào biến ***first***.
- ***Firstminor*** là số bắt đầu cho số hiệu minor trong số hiệu file.
- ***Cnt*** là số lượng số ***minor*** cần thiết.
- ***Name*** là tên device.
- Nếu hàm trả ra giá trị bé hơn 0 thì có nghĩa là đã có lỗi trong khi tạo số hiệu file.
- Khi muốn hủy đăng ký số hiệu ta dùng:

```
unregister_chrdev_region(first, cnt);
```

- Tạo lớp thiết bị
  - Sử dụng hàm

```
cl = class_create(THIS_MODULE, DEVICE_CLASS_NAME)
```

- Hàm trả ra một con trỏ trỏ tới lớp thiết bị vừa tạo
- Hàm trả về ***NULL*** nếu gặp phải lỗi.
- Khi muốn hủy lớp thiết bị ta dùng:

```
class_destroy(cl);
```

- Tạo file thiết bị
  - Sử dụng hàm

```
device_create(cl, NULL, deviceNumber, NULL, DEVICE_NAME)
```

- Hàm sử dụng lớp thiết bị ***cl*** để tạo file thiết bị.
- Hàm sẽ trả về NULL nếu gặp lỗi.
- Khi muốn xóa file thiết bị ta dùng:

```
device_destroy(cl, first);
```

- Định nghĩa các thao tác với file thiết bị:
  - Các thao tác với file thiết bị được gói lại vào một cấu trúc gồm:
    - .open: Thao tác mở file
    - .read: Thao tác đọc file

- .release: Thao tác đóng file

- Mỗi thuộc tính trong cấu trúc trên là một con trỏ trỏ tới hàm xử lý file riêng biệt cho từng thao tác.
- Khởi tạo các thao tác:

```
cdev_init(&c_dev, &fops);
```

- Thêm thiết bị vào hệ thống và khởi động device:

```
cdev_add(&c_dev, first, cnt)
```

- ***cdev_add*** trả về số âm nếu gặp lỗi

3. Hàm xử lý thao tác mở file:

   Vì module này khá đơn giản nên việc mở file ta không phải làm gì ngoài việc in log.
4. Hàm xử lý thao tác đóng file:

   Tương tự thao tác mở file
5. Hàm xử lý thao tác đọc file:

- Hàm nhận vào một ***buffer*** – chuỗi được đưa vào từ ***userspace***, ta sẽ sử dụng nó để đưa kết quả ra ***userspace***.
- Ngoài ra hàm nhận vào ***len*** (length) – độ dài tối đa của ***buffer***.
- Trong trường hợp này, ta sẽ tạo số ngẫu nhiên trong khoảng ***int*** (4 bytes – 32 bit) bằng hàm ***get_random_bytes*** trong thư viện ***<linux/random.h>***.
- Vì output sẽ là một chuỗi nên ta phải chuyển số random thành chuỗi:
  - Nếu số là âm thì ta phải thêm '-' ở trước rồi đảo dấu cho số thành số dương.
  - Lần lượt lấy các số hàng bên phải cho vào chuỗi ta được chuỗi đảo ngược của số random.
  - Lật ngược chuỗi 1 lần nữa ta được chuỗi chuyển từ số random.
- Sử dụng hàm ***copy_to_user*** để copy chuỗi đó từ ***kernelspace*** ra ***userspace***.

6. Chương trình testRandom:

- Trước tiên ta mở file:

- Sau đó đọc file bằng cách truyền 1 ***buffer char[]*** và độ dài tối đa của ***buffer*** vào.
- ***Buffer*** lúc này sẽ là chuỗi số random. Ta có thể dùng hàm ***atoi()*** để chuyển ***buffer*** đó thành số trở lại.

III. Tài liệu tham khảo

Tài liệu hướng dẫn của thầy Phạm Tuấn Sơn

The Linux Kernel documentation — The Linux Kernel documentation