LƯU AN PHÚ

TƯƠNG TÁC VỚI HỆ ĐIỀU HÀNH LINUX

Facebook group: Cùng Nhau Học Linux Kernel

LỜI MỞ ĐẦU

Xuất phát từ nhu cầu thực tiễn của lĩnh vực Linux embedded, chúng tôi là các kỹ sư đang làm việc trong các dự án đó đã quyết định biên soạn ra bộ sách này. Sau khi nghiên cứu chương trình học của các trường đại học nổi tiếng trên thế giới. Bộ sách bao gồm 2 cuốn, trong đó cuốn một sẽ tập trung vào lý thuyết và kỹ thuật lập trình trên tầng user-space, cuốn hai sẽ về kernel-space của hệ điều hành.

Tư tưởng khi chúng tôi viết cuốn sách này là tập trung nhiều về thực hành, ngoài ra các lý thuyết cũng như thực hành đều áp dụng trên các phiên bản OS và hardware mới nhất. Việc này nhằm giúp người học giảm được sự bỡ ngỡ khi bước vào dự án thực tế.

Bộ sách được tham khảo từ 3 cuốn sách nổi tiếng đó là "Advanced programming in the unix environment", "UnderStanding The Linux Kernel" và "Linux Device Drivers".

Trong mỗi một chương sẽ có lý thuyết và bài tập để người học có thể thực hành. Nếu các bạn có thời gian thì nên học và thực hành từ đầu đến cuối. Đó là phương pháp học bài bản và dễ hiểu nhất. Trong trường hợp thời gian không cho phép, các bạn có thể học và thực hành luôn theo cuốn sách thứ 2 của bộ sách này.

Ngoài ra chúng tôi đã tạo ra một group facebook riêng để tập hợp tất cả mọi người trên lãnh thổ Việt Nam, những người có chung niềm đam mê về Linux embedded. Group có tên là "Cùng nhau học Linux kernel", nếu muốn các bạn có thể join vào cùng học hỏi với chúng tôi. Ở đây người mới luôn luôn được chào đón.

Trong quá trình biên soạn không thể tránh được nhiều sai sót. Nếu thấy điểm nào cần cải tiến, các bạn hãy đặt câu hỏi cho chúng tôi thông qua group facebook ở trên hoặc gửi mail cho mình tại địa chỉ: luuanphu@gmail.com.

Tôi muốn gửi lời cảm ơn đến bạn Trương Văn Huy, người đã đóng góp rất nhiều công sức cho bộ sách này.

Ngoài ra tôi cũng gửi lời cảm ơn đến vợ tôi, người đã tạo điều kiện rất nhiều để tôi có thể theo đuổi được đam mê của mình.

Hà Nội, ngày 23 tháng 10 năm 2018

Tác giả

Lưu An Phú

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ CÁC HỆ ĐIỀU HÀNH HỌ UNIX

Ngày nay hệ điều hành đã trở lên quen thuộc với tất cả chúng ta. Tuy nhiên vào những năm 50 của thế kỷ trước, khi đó OS chưa ra đời, người ta phải nạp thẳng code vào máy tính. Mỗi máy tính tại một thời điểm chỉ chạy một chương trình và một chương trình sẽ phải điều khiển toàn bộ máy tính. Với máy tính tại thời điểm đó có kiến trúc đơn giản (không có chuột, bàn phím, màn hình, loa…) nên việc người lập trình viên quản lý toàn bộ máy tính bằng code của mình là khả thi. Tuy nhiên kiến trúc máy tính và yêu cầu tính toàn càng ngày càng phức tạp, do đó người ta cần đến một hệ thống có thể quản lý được máy tính và hỗ trợ nhiều nhất có thể đối với người lập trình viên. Từ yêu cầu thực tế đó hệ điều hành được ra đời.

Các hệ điều hành được ra đời sớm nhất có thể kể đến là GM-NAA I/O, BESYS, SOS, TENEX, Unix… Tuy nhiên thành công nhất chỉ có Unix, nó được thiết kế dựa trên rất nhiều các lý thuyết toán học. Do đó sau nửa thế kỷ trôi qua, phần thiết kế lõi của nó cũng không cần phải chỉnh sửa nhiều. Kiến trúc của Unix được áp dụng cho rất nhiều hệ điều hành phổ biến ngày nay như Android, Window, Linux, MACOS… Và chúng được gọi là các hệ điều hành họ Unix.

Trong chương này, chúng ta sẽ cùng nhau học về các tính chất chung của những hệ điều hành kế thừa từ Unix.

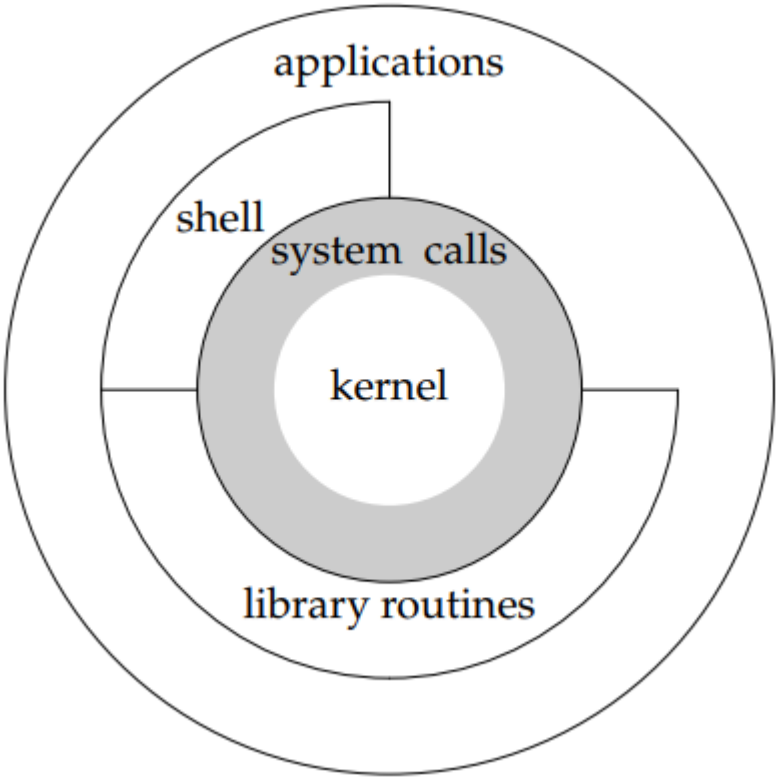
1.1 Các chức năng chính của hệ điều hành

Là lớp vỏ bảo vệ cho hardware của hệ thống: Hiểu một cách đơn giản thì hardware của hệ thống giống như lòng trứng. Để tương tác trực tiếp với chúng thì người lập trình viên phải cẩn thận và hiểu rõ phần cứng. Tuy nhiên khi có hệ điều hành thì việc đó sẽ không cần thiết nữa. Hệ điều hành sẽ tạo ra lớp vỏ trứng bao quanh toàn bộ phần cứng. Lúc này lập trình viên thay vì tương tác với phần cứng thì sẽ tương tác với lớp vỏ là hệ điều hành, sau đó hệ điều hành sẽ là người làm việc với phần cứng.

Là đối tượng duy nhất sở hữu, quản lý và phân phối phần cứng trong hệ thống: Khi hệ thống đi vào hoạt động sẽ có rất nhiều đối tượng tồn tại trong nó – Ví dụ như các chương trình Word, Excel, Chrome, … Và hệ điều hành cũng là một đối tượng nằm trong số đó. Tuy nhiên khác với các đối tượng còn lại, hệ điều hành là đối tượng được khởi động đầu tiên trong hệ thống, nó khởi tạo toàn bộ phần cứng và chiếm luôn quyền sở hữu chúng. Sau đó nó sẽ khởi tạo các đối tượng còn lại và quản lý, phân phối phần cứng cho toàn hệ thống.

Cung cấp môi trường hoạt động và xử lý xung đột giữa các đối tượng: Do hệ điều hành là đối tượng đầu tiên được tạo ra trong hệ thống. Sau đó tất cả các đối tượng còn lại đều được sinh ra bởi hệ điều hành, do đó nó có toàn quyền điều khiển các đối tượng còn lại. Nó có thể sinh ra một đối tượng mới, tạm dừng một đối tượng đang chạy hoặc kết thúc vòng đời của chúng. Mỗi khi trong hệ thống xuất hiện trạng thái xung đột giữa các đối tượng thì hệ điều hành sẽ đứng ra phân xử và nó sẽ trực tiếp thi hành quyết định của mình. Tất cả các đối tượng còn lại đều phải tuân theo quyết định của nó.

1.2 Kiến trúc của hệ điều hành



Hình 1: Kiến trúc của hệ điều hành họ Unix

Có rất nhiều cách để phân chia các lớp trong hệ điều hành, không có cách nào là chính xác hoàn toàn. Việc phân chia hệ điều hành ra thành những lớp nào đều phụ thuộc vào từng góc nhìn khác nhau. Trong phạm vi cuốn sách này, chúng ta sẽ coi hệ điều hành gồm 4 lớp cơ bản như sau:

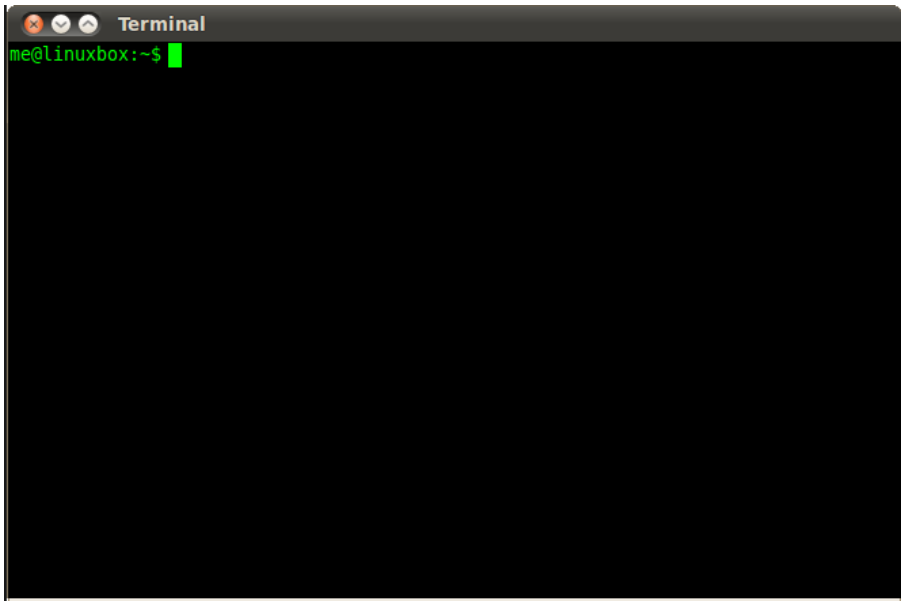
+ Lớp kernel: Đây là lớp trong cùng, nó bao ngoài phần cứng, quản lý và cung cấp những chức năng cơ bản của hệ điều hành như: Lập lịch, quản lý bộ nhớ, quản lý ngắt,…

+ Lớp system call: Do cách thiết kế hệ điều hành không cho phép các ứng dụng từ tầng application được phép truy cập thẳng vào lớp kernel (Để tránh 1 lỗi trên tầng application có thể làm sập hệ thống). Nên họ đã thiết kế 1 lớp để ngăn cách gọi là lớp system call. Nhiệm vụ của lớp system call là cung cấp các đầu hàm (Ví dụ như read(), write()) cho lớp application sử dụng.

+ Lớp application: Đây là lớp ngoài cùng của hệ điều hành, là nơi tương tác với user. Các tiến trình như word, excel... mà user sử dụng đều được chạy ở lớp này.

1.3 Giao diện người sử dụng

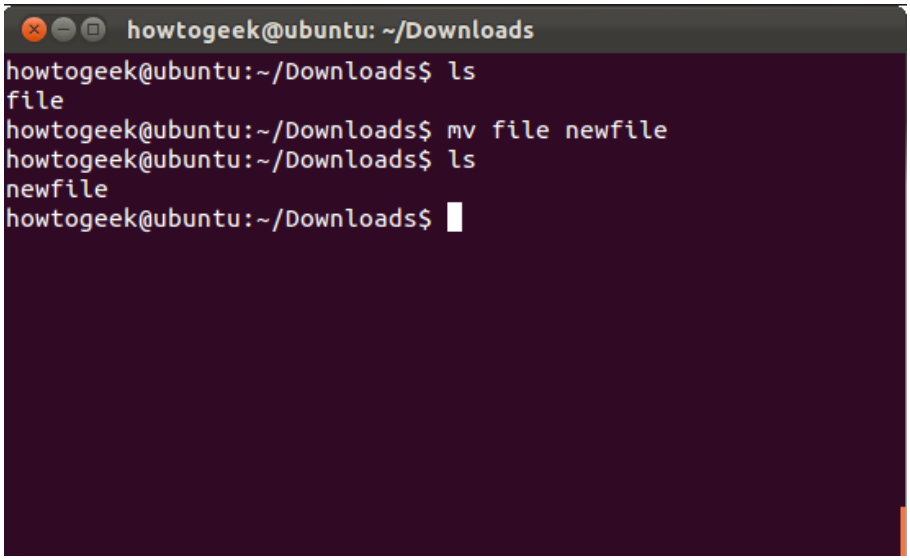
Khác với hệ điều hành như Window – hầu hết sự tương tác diễn ra thông qua giao diện cửa sổ. Trong các hệ điều hành như Linux thì giao diện dòng lệnh mới là công cụ đủ mạnh để tương tác với hệ điều hành. Có rất nhiều công việc bạn không thể sử dụng giao diện cửa sổ để xử lý mà bắt buộc phải sử dụng dòng lệnh. Dưới đây là hình ảnh giao diện bằng dòng lệnh của Linux:



Hình 2: Giao diện dòng lệnh của Linux

Trong các hệ thống embedded người ta sẽ disable chế độ graphic, do đó khi boot lên màn hình chỉ có duy nhất cửa sổ command line được hiện lên.

Các câu lệnh sẽ tuân theo cấu trúc như sau:



Hình 3: Ví dụ một câu lệnh trong Linux

Trong đó phần đầu câu lệnh (mv) sẽ là tên của chương trình thực hiện câu lệnh đó. Bản chất của tất cả các câu lệnh là các chương trình được đặt trong một thực mục nhất định của hệ thống. Thông thường sẽ là thư mục /bin và /sbin. Khi chúng ta gọi một câu lệnh thì hệ thống sẽ run chương trình trùng tên nằm trong 2 thư mục đó. Trong trường hợp không có chương trình nào trùng tên hệ thống sẽ báo lỗi.

Phần tham số theo sau tên câu lệnh (file new file) là các chuỗi string được truyền cho chương trình thực hiện câu lệnh. Một lưu ý với các bạn là tất cả các tham số ở dạng chữ hoặc số khi truyền vào cho chương trình đều chuyển về dạng string.

Một số câu lệnh khi thực hiện cần những quyền nhất định. Ví dụ khi tương tác với các file hệ thống chúng cần quyền của người quản trị… Các quyền của một câu lệnh sẽ

phụ thuộc và user nào đang thực hiện câu lệnh đó. Do vậy trong một số trường hợp, chúng ta phải tiến hành chuyển đổi user để cấp quyền cao hơn cho câu lệnh đó.

1.4 Chương trình và tiến trình

Chúng ta sẽ đi tiếp với một khái niệm quan trọng của hệ điều hành – chương trình và tiến trình.

+ Chương trình: Là các file binary được build từ source code. Chúng nằm trên ổ cứng và không tương tác cũng như sử dụng bất cứ một tài nguyên nào của hệ thống. Do đó cho dù hệ thống có lưu trữ bao nhiêu chương trình thì hiệu năng của nó cũng sẽ không bị giảm đi.

+ Tiến trình: Là những chương trình đã được load vào hệ thống. Do đó chúng sẽ tương tác và sử dụng tài nguyên của hệ thống. Càng nhiều tiến trình chạy trong hệ thống thì hiệu năng sẽ càng bị giảm đi.

Cũng giống như việc đặt tên để định danh cho con người, hệ điều hành sẽ đánh số cho từng tiến trình để định danh chúng. Số định danh đó sẽ là số thứ tự mà tiến trình đó được load vào hệ thống. Chúng được gọi là các process id. Hệ thống sẽ tương tác với các tiến trình thông qua định danh của chúng – process id.

Input và output của tiến trình: Chúng là 2 file với file đầu là nơi tiến trình sẽ đọc dữ liệu đầu vào cho các hàm như scanf() và file thứ 2 sẽ là nơi tiến trình ghi kết quả đầu ra trong các hàm như printf(). Thông thường file input sẽ là bàn phím và file output sẽ là màn hình console.

1.5 Unix là một hệ điều hành đa nhiệm

Khả năng đa nhiệm của hệ điều hành Unix nhằm mục đích tạo cho người dùng cảm giác hệ thống đang xử lý nhiều công việc một lúc, đây cũng là một tính năng làm nên tên tuổi của Unix so với những đàn anh đi trước. Đối với con người 1 2ms là rất ngắn và không thể cảm nhận được, tuy nhiên đối với máy tính thì khoảng thời gian đó đủ để làm nhiều công việc khác. Do vậy hệ thống liên tục chuyển đổi giữa các công việc khác nhau nhưng vẫn đảm bảo khả năng xử lý tức thì cho người dung.

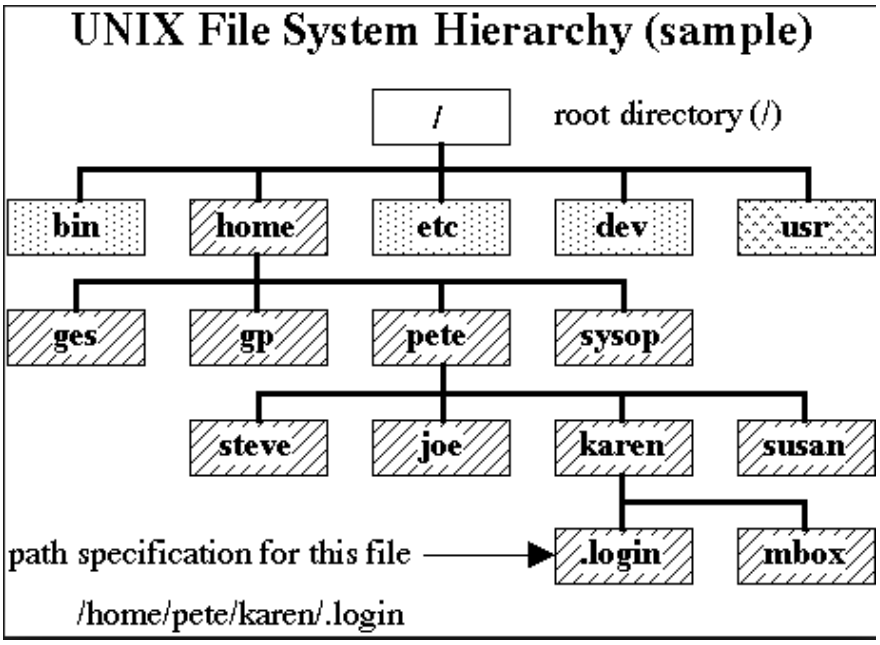
Lấy ví dụ một người dùng đang vừa soạn thảo văn bản vừa nghe nhạc. Chương trình soạn thảo văn bản cần 1 ms để xử lý mỗi khi người dùng gõ 1 phím bất kỳ. Chương trình nghe nhạc cần chạy định kỳ 1ms mỗi 1s. Nếu hệ thống không chuyển đổi giữa các task 1 cách hợp lý thì đôi khi chương trình nghe nhạc có thể bị gián đoạn do tại thời điểm đó CPU đang chạy chương trình soạn thảo văn bản. Tuy nhiên hệ điều hành sẽ không để cho điều đó xảy ra. Nó sẽ ưu tiên cho chương trình nghe nhạc được chạy đúng thời điểm, nếu tại thời điểm đó user ấn 1 phím bất kỳ, nó sẽ không xử lý luôn mà sẽ delay 1 2ms để chương trình nghe nhạc chạy xong rồi mới xử lý việc đó. Tuy nhiên việc delay 1 2ms trong soạn thảo văn bản sẽ không gây cảm giác xử lý trễ đối với người dùng và anh ta sẽ có giảm giác hệ thống chạy đồng thời cả chương trình nghe nhạc và soạn thảo văn bản.

1.6 Hệ thống file system

Các hệ điều hành trước Unix đã có hệ thống file system. Tuy nhiên đến lượt mình thì Unix nâng cấp chúng thêm một bậc nữa. Hệ thống sẽ coi toàn bộ các đối tượng tồn tại trong nó đều là file. Các đối tượng đó có thể là các hardware device, process, user… Từ đó hệ thống có thể quản lý tất cả các đối tượng thông qua một phương thức duy nhất đó là tương tác qua file.

Trong Unix file có thể hiểu như một định danh vì nhiều khi nó đại diện cho dữ liệu nằm trên ổ cứng hoặc một thiết bị nào đó. Ví dụ như các file đại diện cho từng process chúng nằm trong thư mục /proc/process_id. Mỗi một file sẽ có các thuộc tính ví dụ như kích thước, quyền sở hữu, ngày chỉnh sửa … Ngoài ra có 1 file dạng đặc biệt đó là thư mục. Thư mục là một file những dữ liệu bên trong nó chính là danh sách tên của các file nằm trong nó.

Việc tổ chức các file vào trong các thư mục và tạo các thư mục con trong thư mục cha nhằm phân cấp và sắp xếp hệ thống file người ta còn gọi chúng với cái tên cây thư mục. Cây thư mục có các node lá là file, node cành là các thư mục vào node gốc là thư mục root của hệ thống.



Hình 4: Cấu trúc của một cây thư mục

1.7 Bài tập

Bài 1: Hướng dẫn build chương trình trong Linux dùng gcc

Trong Ubuntu command line chúng ta dùng câu lệnh

```
$ sudo apt-get install build-essential
```

Ví dụ: để build và chạy một chương trình hello.c đơn giản trêm command line của Ubuntu:

```
#include <stdio.h>
int main()
{
    printf("Hello, World!");
    return 0;
}
```

```
$ gcc -o hello hello.c
```

```
$ ./hello
```

```
Hello, World!
```

Bài 2: Ví dụ về in một file ra cửa sổ console – readfile.c:

```c
#include <stdio.h>
#include <stdlib.h> // For exit()

int main()
{
    FILE *fptr;
    char filename = "/etc/passwd";
    char c;

    // Open file
    fptr = fopen(filename, "r");
    if (fptr == NULL)
    {
        printf("Cannot open file \n");
        exit(0);
    }

    // Read contents from file
    c = fgetc(fptr);
    while (c != EOF)
    {
        printf("%c", c);
        c = fgetc(fptr);
    }

    fclose(fptr);
    return 0;

}
```

Build và chạy chương trình bằng command line:

$ gcc -o readfile readfile.c

$ ./readfile

Bài 3: Chương trình in ra tên của process từ process id nhập từ bàn phím:

```c
#include <stdio.h>
#include <stdlib.h> // For exit()

int main()
{
    FILE *fptr;
    char processpath[100], c;
    int id;

    printf("Xin hay nhap process id = ");
    scanf("%d", &id);

    sprintf(processpath, "/proc/%d/cmdline", id);
    fptr = fopen(processpath, "r");
    if (fptr == NULL)
    {
        printf("Khong co process id = %d \n", id);
        exit(0);
    }

    c = fgetc(fptr);
    while (c != EOF)
    {
        printf("%c", c);
        c = fgetc(fptr);
    }
```

```
    fclose(fptr);
    return 0;
      }
```

Bài 4: Viết 1 chương trình có khả năng hiện thị nội dung của 1 file lên console giống với lệnh cat. Đặt tên nó là MyCat và copy vào thư mục bin. Sau đó sử dụng nó giống như lệnh cat của hệ thống.

Bài 5: Viết 1 chương trình C có khả năng in ra tọa độ của chuột trên máy tính. Ví dụ: X/Y. Trong đó X, Y là hoàn độ và tung độ của con trỏ chuột trên màn hình.