Giới thiệu về File I/O

Giới thiệu

Triết lý của các hệ điều hành Unix và Linux là mọi thứ đều là file (Everything is a file), có nghĩa là mọi thứ trong hệ thống đều có thể được biểu diễn dưới dạng một file, từ file cổ điển .txt đến các đường dẫn (directory) hay các thiết bị như character device, socket.... Khi chúng ta sử dụng command "ls -la" để liệt kê các file trong đường dẫn hiện tại, ký tự đầu tiên của từng dòng chính là loại của file đó. Hệ điều hành Linux có các loại file sau đây:

Ký hiệu	Loại file
-	File thường (regular file)
d	Đường dẫn (directory)
c	Character device file
b	Block device file
s	Domain socket
p	pipe
l	symbolic link

Trong chương này, chúng ta sẽ tìm hiểu về cách truy cập file (file I/O), các system call cần thiết và cách tương tác với các file.

Để đọc hoặc ghi vào một file, trước tiên nó phải được mở ra. Mỗi file được mở sẽ được định danh bởi một số nguyên dương duy nhất được gọi là một mô tả file (file descriptor). Các system call thao tác với file này sẽ dùng file descriptor này là tham số truyền xuống kernel. Với mỗi process, kernel lưu và duy trì một bảng danh sách các mô tả file được gọi là *file table* và sử dụng bảng đó để tra cứu theo file descriptor và tìm đến đúng file mà tiến trình tầng user muốn thao tác.

Theo quy ước, một chương trình khi chạy luôn có 3 file được mở có số mô tả file lần lượt là 0 (standard input - stdin), 1 (standart output - stdout) và 2 (standard error - stderr). Khi lập trình, chúng ta không nên hard code các file descriptor bằng các số đó mà nên sử dụng các macro (định nghĩa trong thư viện POSIX <unistd.h>) là STDIN_FILENO, STDOUT_FILENO và STDERR_FILENO. Thông thường, stdin được kết nối với bàn phím trong khi stdout và stderr được kêt nối với màn hình của thiết bị.

Chúng ta cũng có thể đổi hướng (redirect) những file description này hoặc thậm chí lái output của một chương trình này vào làm input của một chương trình khác. Trên thực tế, đây là việc chúng ta làm khá thường xuyên trong khi lập trình Linux mà không hề hay biết, đặc biệt là trong các thao tác hay lời gọi của shell. Ví dụ khi chúng ta muốn liệt kê các tiến trình đang chạy của hệ thống (bằng câu lệnh “ps aux”) và muốn lưu kết quả vào một file để dễ đọc hơn hoặc lưu

trữ cho việc dùng sau này, có thể dùng câu lệnh sau:

```
ps  aux > vimentor.txt
```

Với câu lệnh “ps aux”, các tiến trình đang chạy trên hệ thống sẽ được hiển thị lên màn hình (stdout), nhưng với câu lệnh trên, kết quả của câu lệnh “ps aux” đã được đổi hướng làm input (stdin) của file vimentor.txt. Khi mở file vimentor.txt, các bạn sẽ thấy nó chính là kết quả đáng lẽ hiển thị ra màn hình của câu lệnh “ps aux” đó.

Mô hình Universality I/O

Một trong những tính năng khác biệt của hệ điều hành Linux là mô hình universality I/O. Có nghĩa bốn các system call cơ bản thao tác file open(), read(), write() và close() có thể đươc sử dụng để thao tác Input/Output không chỉ cho các file thông thường mà cho tất cả các file, bao gồm cả các device file (trong đường dẫn /dev/) hay các terminal (/dev/tty).

Mô hình universal I/O trong Linux được thực hiện bằng việc quy định các lập trình viên khi tạo ra các file system và device driver đều phải tạo ra các system call Input/Output. Ví dụ, khi viết một device driver, chúng ta phải tạo các device file và các entry point (open, read, write, release) như hướng dẫn trong bài Entry point của device driver. Mô hình universal I/O giúp cho việc lập trình các ứng dụng Linux đơn giản hơn rất nhiều vì chúng ta không cần quan tâm đến cách kernel thao tác với các device hay terminal đó mà chỉ cần dùng các system call thân thiện là open(), read(), write() và close(). Trong trường hợp người dùng muốn thực hiện thao tác ngoại lệ ngoài bốn system call trên thì có thể sử dụng system call ioctl() - chúng ta sẽ nói rõ hơn về system call này ở một bài khác.

Kết luận

Triết lý mọi thứ đều là file của Unix và Linux rõ ràng là rất khác biệt so với hệ điều hành windows mà chúng ta hay dùng. Mô tả file (file descriptor) định danh cho một file là chìa khóa cho triết lý này, và cho việc lập trình với các file trên Linux. Các bài tiếp theo chúng ta sẽ học cách thao tác các file trong Linux.