# Các khái niệm cơ bản về Linux

Có 3 nền tảng chính trong lập trình hệ thống trên Linux: các lời gọi hệ thống, thư viện C và trình dịch C.

## Các lời gọi hệ thống

Lập trình hệ thống bắt đầu và kết thúc với các lời gọi hệ thống. Lời gọi hệ thống (Thường viết tắt là syscalls) là các lời gọi hàm được thực hiện từ user space (không gian người dùng) – trình sửa văn bản, các trò chơi điện tử và v..v. – tới kernel (nhân – lõi bên trong của hệ thống) để mà yêu cầu một vài dịch vụ hoặc tài nguyên từ hệ điều hành.

Linux cung cấp ít lời gọi hệ thống hơn các nhân hệ điều hành khác. Ví dụ, số lượng lời gọi hệ thống của kiến trúc x86-64 vào khoảng 300, so sánh với hàng ngàn lời gọi hệ thống trên Microsoft Windows. Trong nhân Linux, mỗi kiến trúc máy tính ( như Alpha, x86-64 hoặc PowerPC) có thể gia tăng lời gọi hàm chuẩn với bản thân nó. Do đó, các lời gọi hệ thống khả dụng ở một kiến trúc có thể khác so với các kiến trúc khác. Tuy nhiên, một tập con rất lớn các lời gọi hệ thống – lớn hơn 90 % - được áp dụng tới tất cả các kiến trúc.

### 1.1. Các lời gọi hệ thống không gọi được trực tiếp

Nó không liên kết trực tiếp các ứng dụng người dùng với nhân hệ điều hành. Vì lí do bảo mật và độ tin cậy, các ứng dụng người dùng không được cho phép thực hiện các đoạn mã hoặc dữ liệu ở trong kernel này. Thay vào đó, kernel phải cung cấp các cơ chế mà dựa vào đó ứng dụng người dùng có thể báo hiệu cho kernel biết rằng nó mong muốn một lời gọi hệ thống được thực hiện. Đối với các kiến trúc khác nhau thì sẽ có cơ chế khác nhau. Ở trên kiến trúc i386, ví dụ, một ứng dụng người dùng thực hiện một chỉ thị lời ngắt phần mềm, int, với giá trị là 0x80. Chỉ chị này là nguyên nhân có sự thay đổi trong kernel, nơi mà kernel thực thi một trình điều khiển ngắt phần mềm với giá trị 0x80.

Chương trình nói với kernel cái mà lời gọi hệ thống thực hiện với gái trị tham số thông qua các thanh ghi (machine registers). Các lời gọi hệ thống được đánh dấu bởi các giá trị số bắt đầu từ 0. Trong kiến trúc i386, một lời gọi hệ thống với giá trị 5 , ứng dụng người dùng đẩy giá trị 5 vào thanh ghi eax trước ghi khi thực hiện chỉ thị *int*.

Tham số truyền vào được điều khiển theo các tương tự. Trong i386, các thanh ghi được sử dụng cho các tham số có thể là eax, ebx, ecx, esi và edi tương đương với 5 tham số đầu tiên. Các lời gọi hệ thống lớn có tham số lớn hơn 5 thường hiếm xảy ra và một thanh ghi đơn được sử dụng để trỏ lời bộ đệm (buffer) trong không gian người dùng nơi mà tất cả các tham số được giữ ở trong đó. Dĩ nhiên, đa số lời gọi hệ thống thường chỉ có một cặp tham số.

Các kiến trúc khác điều khiển lời gọi hệ thống theo các khác nhau, mặc dù về bản chất là giống nhau.

### 1.2. Thư viện C

Thư viện C (libc) là trái tim trong các ứng dụng Unix. Nó cung cấp các dịch vụ lõi và các lời gọi hệ thống. Trên các hệ thống Linux hiện đại, thư viện C được cung cấp bởi GNU *libc,* viết tắt là glibc.

Trong thư viện GNU C, ngoài việc cung cấp các thư viện C chuẩn, glibc còn cung cấp các lời gọi hệ thống hoàn thiện, hỗ trợ đa luồng (threading support) và các nền tàng cho các ứng dụng cơ bản.

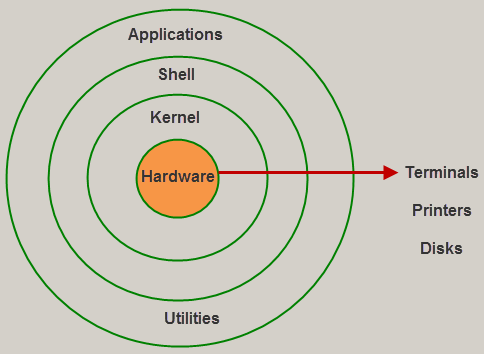
### 1.3. Trình dịch C

Trong Linux, Trình dịch C chuẩn được cung cấp bởi GNU Compiler Collection (gcc). Nguyên bản, gcc là phiên bản GNU của cc, C Compiler. Do đó, gcc là viết của GNU C Compiler.

# LINUX Kernel

## Định nghĩa

Kernel là một chương trình máy tính mà là phần lõi của hệ điều hành, với quyền điều khiển mọi thứ ở trong hệ thống. Trong đa phần các hệ thống, Kernel là chương trình đầu tiên được đọc trong trình khởi động (start-up) ngay sau bootloader. Nó chịu trách nhiệm tương tác với các ứng dụng người dùng được chạy ở trong “user mode” và các phần cứng vật lý.



Nó điều khiển các phần còn lại của chương trình khởi động cũng như các yêu cầu vào ra từ phần mềm, dịch chúng thành các chỉ thị xử lý dữ liệu đối với CPU. Nó điều khiển bộ nhớ và các thiết bị ngoại vi như bàn phím, màn hình, máy in, loa v..v..

## Các loại Kernel

Nhìn chung đa phần kernel được chia thành 3 loại:

* Monolithic
* Microkernel
* Hybrid

Linux là một Monolithic Kernel. Trong khi OS X và Windows 7 sử dụng hybrid kernel.

### Microkernel

Sử dụng cách tiếp cận chỉ quản lý những cái nó có: CPU, bộ nhớ và IPC. Mọi thứ còn lại trong máy tính tính có thể xem là phụ kiện và có thể điều khiển trong user mode. Microkernel có một thuận lợi là tính di động bởi vì chúng ta không phải lo nếu bạn thay đổi video card hoặc thậm chí là hệ điều hành của bạn miễn là hệ điều hành vẫn truy cập vào phần cứng với cùng cách.

Microkernel có phạm vi rất nhỏ cho cả bộ nhớ và khoảng không được cài đặt, và chúng hướng tới bảo mật hơn bởi vì chỉ các tiến trình xác định chạy trong “user mode”, cái mà không có quyền cao như “supervisor mode”.

Ưu điểm:

* Di động
* Phạm vi cài đặt nhỏ
* Phạm vi bộ nhớ nhỏ
* Bảo mật

Nhược điểm:

* Phần cứng trừu tường hơn thông qua các driver
* Phần cứng tương tác chậm hơn vì các drivers ở trong user mode
* Các tiến trình phải đợi ở trong 1 hàng đợi để lấy thông tin
* Các tiến trình không thể truy cập tới tiến trình khác mà không phải chờ đợi

### Monolithic Kernel

Monolithic kernel thì ngược lại với microkernel bởi vì chúng bao gồm không chỉ CPU, bộ nhớ và IPC mà chúng cũng bao gồm các device driver, quản lý file system và system server call. Monolithic kernel hướng tới việc truy cập vào phần cứng và đa nhiệm tốt hơn bởi vì nếu một chương trình cần lấy thông tin từ bộ nhớ hoặc tiến trình khác chạy thì có một đường kết nối trực tiếp tới nó và không phải đợi trong hàng đợi và chờ hoàn thành. Điều này tuy nhiên là nguyên nhân bởi vì càng nhiều thứ chạy ở trong “supervisor mode”, thì càng nhiều điều có thể làm hệ thống xảy ra lỗi.

Ưu điểm:

* Truy cập trực tiếp tới phần cứng đối với chương trình
* Dễ dàng đối với các tiến trình để giao tiếp với nhau
* Nếu thiết bị của bạn được hỗ trợ thì hệ thống sẽ hoạt động mà không cần cài đặt thêm
* Các tiến trình tương tác nhanh hơn bởi vì không có hàng đợi trong thời gian xử lý

Nhược điểm:

* Cài đặt trong phạm vi rộng
* Cần nhiều bộ nhớ để cài đặt
* Ít bảo mật hơn bởi vì mọi thứ đều chạy trong quyền supervisor

### Hybrid Kernel

Hybrid kernels có thể lựa chọn cái mà chúng muốn chạy trong “user mode” và cái mà chạy trong “supervisor mode”. Thông thường, device driver, filesystem I/O sẽ chạy trong user mode trong khi IPC và server call sẽ được giữ trong “supervisor mode”. Điều này mang lại nhưng ưu điểm tốt nhất của cả hai loại trên nhưng thường yêu cầu nhiều công việc với các nhà sản xuất phần cứng bởi vì tất cả driver chịu trách nhiệm sử dụng chúng. Nó cũng có một vài vấn đề với độ trễ mà được kế thừa từ microkernel.

Ưu điểm:

* Nhà phát triển có thể lựa chọn cái mà được chạy trong user mode và cái được chạy trong supervisor mode
* Phạm vi cài đặt nhỏ hơn monolithic kernel
* Mềm dẻo hơn các mô hình khác

Nhược điểm:

* Có thể gặp chung tình trạng giật như microkernel
* Các device driver cần được quản lý bởi user