# Biên dịch nhân Linux

Tác giả: Hoàng Ngọc Diêu

# Mục lục

# 1 Tổng quan về nhân Linux trên phương diện biên dịch lại

## 1.1 Nhân Linux và việc biên dịch lại nhân

Nhân Linux là một "nhân hiện đại" có tính module rất cao. Từ kernel phiên bản 2.6.x trở đi, có rất nhiều chức năng và mở rộng. Với tinh thần "biên dịch nhân", một yếu tố chính yếu và quan trọng nhất cần ghi nhận đó là tính phân bộ (modularity) của nhân Linux.

Đối với người dùng bình thường, modularity cho phép chọn lựa cách biên dịch các drivers của nhân theo dạng modules hay theo dạng biên dịch trực tiếp vào nhân. Thông thường, khi xác lập cấu hình cho nhân có ba chọn lựa: Y, M và N.

Có những "driver" không thể biên dịch như một module vì nó phải được load and link trực tiếp ngay khi nhân khởi động. Cũng có những "driver" cho phép chọn như một module và được tải trong khi và sau khi nhân được khởi động. Điểm chính yếu cần nắm bắt trong giới hạn chủ đề "Biên dịch nhân Linux" là hiểu rõ tại sao phải chọn M (cho module), Y (cho biên dịch trực tiếp) và N (không dùng) các *drivers* này.

- Biên dịch trực tiếp vào kernel có nghĩa là các "*drivers*" này dù có được dùng hay không vẫn được tải lên khi nhân khởi động và tất nhiên nó sẽ chiếm một phần bộ nhớ. Lợi điểm chính của chọn lựa này là một khi "*drivers*" đã được biên dịch vào nhân thì không còn phải quan ngại đến tính trung thực của nhân và các *driver* nữa. Các hệ thống làm việc đòi hỏi tính bảo mật cao không dùng *modules* mà biên dịch thẳng vào nhânkernel để tránh trường hợp các modules không tin cậy "bị" cài vào nhân lúc nào đó trong quá trình hoạt động của máy. Lợi điểm kế tiếp của chọn lựa này là tính hiệu xuấtxuất (rất nhỏ), khi cần driver thì đã có sẵn và không cần ứng tải nữa.

- Biên dịch như các *modules* cho nhân có nghĩa là chỉ khi nào cần dùng các "*drivers*" này mới được ứng tải. Lợi điểm của chọn lựa này nổi bật ở khía cạnh xử dụng bộ nhớ và tài nguyên trên máy. Với lựa chọn này, bạn có thể tạo nên một nhân rất nhỏ và dễ dàng di chuyển cho nhiều mục đích khác nhau. Lợi điểm kế tiếp là khả năng biên dịch lại chỉ một hoặc một số *modules* nào đó (cần cập nhật chẳng hạn). Tất nhiên để thực hiện chuyện này thì phải thoả mãn tất cả những đòi hỏi về tính phụ thuộc cho hệ thống.

## 1.2 Tóm tắt các bước biên dịch (dành cho những ai thiếu kiên nhẫn)

Biên dịch nhân Linux rất đơn giản nếu như đã hiểu rõ các quy trình và các bước thực hiện. Sau đây là các lệnh cần thiết, giả định bạn đã có trọn bộ các công cụ cần thiết để biên dịch:

Chuyển vào thư mục `/usr/src`, nơi thông thường chứa mã nguồn để biên dịch nhân:

```
$ cd /usr/src
```

`<KERNEL_SRC>` là phiên bản kernel cần biên dịch, ví dụ ở đây tải mã nguồn được nén ở dạng bz2)

```
$ wget http://www.kernel.org/pub/linux/kern...>.tar.bz2 (<KERNEL_SRC>
```

Xác thực chữ ký và thực tính của mã nguồn:

```
$ gpg --verify <KERNEL_SRC>.tar.bz2.sign <KERNEL_SRC>.tar.bz2
```

Xả nén gói chứa mã nguồn:

```
$ bzip2 -dc <KERNEL_SRC>.tar.bz2 | tar xvf -
```

Nếu muốn dùng giao diện đồ họa để điều chỉnh các chọn lựa cho cấu hình nhân thì dùng lệnh:

```
$ make xconfig
```

Tập hợp lệnh dùng để tạo các file phụ thuộc và các file bao gồm (include), tiếp theo là dọn dẹp các *objects* không cần thiết và biên dịch nhân ở dạng nén:

```
$ make dep clean bzImage
```

Biên dịch các modules đã được chọn lựa bằng lệnh:

```
$ make modules
```

Chchuyển sang chế độ super user cho lệnh tiếp theo:

```
$ su
```

Cài các *modules* vào thư mục `/lib/modules/<KERNERL_SRC>` với quyền của super user:

```
# make modules_install
```

Cài đặt nhân và các file cần thiết vào thư mục `/boot`:

```
# make install
```

Bước thứ 10 có thể thay thế bằng một loạt thao tác bằng tay (chi tiết ở phần 9.2) nếu như bản phân phối[1] Linux không có sẵn một số công cụ thuộc gói `mkinitrd` (chỉ có trong bản *RedHat* và các bản dựa trên RedHat).

[1] distribution

**Lưu ý**: đối với  loạt nhân 2.6.x, bạn có thể dùng các bước như trên. Tuy nhiên bước "`make dep`" không cần thiết nữa. Bài viết này bao gồm cho cả phiên bản 2.4.x và 2.6.x nên có một số chi tiết không cần thiết cho kernel 2.6.x. Tuy vậy, những chi tiết này sẽ không ảnh hưởng đến sự thành công của quy trình biên dịch nhân.

# 2 Tại sao cần biên dịch lại nhân Linux?

Đối với người dùng đã quen với những hệ điều hành "đóng" thì khái niệm biên dịch lại nhân là một khái niệm hết sức lạ lẫm. Điều này cũng dễ hiểu vì kernel của các hệ điều hành "đóng" hiển nhiên là "đóng" và người dùng bình thường không thể có cơ hội tiếp cận với mã nguồn của nhân để có thể biên dịch lại nhân nếu muốn. Trong khi đó, mã nguồn của nhân Linux hoàn toàn "mở" và đây là điều kiện rất thuận lợi cho vấn đề biên dịch lại nhân. Câu hỏi được đặt ra là tại sao lại cần phải biên dịch lại nhân Linux ?

**Câu trả lời ngắn**: không cần nếu như không cần và cần nếu như cần :)

**Câu trả lời dài**: có vô số lý do khiến cho người dùng cần phải biên dịch lại nhân Linux. Sau đây là một số trường hợp thường gặp nhất:

a. tái biên dịch kernel để chữa lỗi của nhân. Nếu các lỗi này thuộc về lõi của nhân thì phải vá nguồn của nhân và biên dịch lại nó để sửa chữa các lỗi được công bố.

b. biên dịch lại nhân để nâng cao hiệu năng của nhân. Theo mặc định, các bản phân phối Linux thường kèm một phiên bản nhân được biên dịch với hầu hết những thành phần có sẵn để có thể đáp ứng rộng rãi cấu hình phần cứng (có thể hiện diện trên các máy). Đây là điểm lợi tổng quát lúc khởi điểm. Tuy nhiên, sau khi đã cài thành công và nắm chắc máy có những thiết bị gì (*sound card, graphic card, network cards, SCSI card*..... ) và biết rõ cần những thành phần nào cho cấu hình của máy thì không có lý do gì phải bao gồm trọn bộ các thứ không cần thiết và không dùng. Đối với nhân 2.4.x, mức độ nâng cao hiệu năng không rõ rệt (ngoại trừ dùng phương pháp *test load* để đo). Tuy nhiên, từ phiên bản 2.6.x trở đi, việc biên dịch lại và điều chỉnh "*driver*" cho nhân tạo hiệu xuất rõ rệt, nhất là trong việc điều chỉnh "thời biểu" (*scheduling*) của các công tác mà hệ thống phải đảm nhiệm.

c. biên dịch lại nhân để loại bỏ những "*drivers*" không được dùng và có thể gây "hiểu lầm" cho nhân, tạo ra trường hợp máy có những triệu chứng hoạt động thiếu ổn định và hay gây lỗi.

d. biên dịch lại nhân để thử nghiệm một chức năng hoặc một *module* mình vừa tạo ra. Trường hợp này không nhiều như các trường hợp trên nhưng cũng nằm trong các lý do phổ biến.

# 3 Cấu trúc và quy ước số hiệu phiên bản của nhân Linux

Phiên bản của nhân Linux có quy ước rất đơn giản và dễ nhớ. Vấn đề này cần nắm rõ trước khi chọn một phiên bản nào đó của nhân Linux để vá và biên dịch.

Phiên bản của nhân Linux bao gồm ba nhóm số tách ra bởi các dấu chấm. Ví dụ: **2.4.26**

*Số thứ nhất: 2* là số hiệu phiên bản chính

*Số thứ nhì: 4* là chỉ định cho tình trạng phiên bản. Nếu số này là số chẵn, nó chỉ định cho phiên bản ổn định (stable), có thể dùng cho môi trường *production*. Nếu số này là số lẻ, nó chỉ định cho phiên bản không ổn định, nó thường dùng trong môi trường đang phát triển (*development*). Các kernel thuộc dạng này thường có nhiều lỗi và không ổn định. Nếu dùng các phiên bản này để tìm lỗi và thông báo cho nhóm phát triển nhân Linux thì đây là điều rất tốt. Không nên dùng phiên bản phát triển cho môi trường *production*.

*Số thứ ba: 26* là chỉ định cho số hiệu phát hành của một phiên bản nhân Linux. Một phiên bản ổn định của một nhân Linux có thể có nhiều số hiệu phát hành khác nhau.

Đây là các quy ước chung cho dạng nhân Linux "*vanilla*" có nghĩa là ứng dụng cho các phiên bản nhân từ `http://www.kernel.org`, các phiên bản nhân được điều chỉnh bởi mỗi bản phân phối có những điểm dị biệt. Có nhiều bản Linux xử dụng số hiệu con[2] cho phiên bản nhân họ đã điều chỉnh. Ví dụ RedHat có những cập nhật phụ cho các kernel như: 2.4.20-8 chẳng hạn. Điều cần nắm ở đây là chỉ nên xử dụng phiên bản ổn định (*stable*) của nhân Linux (số chẵn ở giữa) cho môi trường *production* và dùng phiên bản thử nghiệm[3] của nhân Linux (số lẻ) cho môi trường thử nghiệm và phát triển.

# 4 Đòi hỏi tối thiểu trong việc biên dịch lại nhân Linux

Trước khi bắt tay vào việc biên dịch lại nhân Linux, điều cần thiết là phải có đủ chỗ chứa trên đĩa. Ít nhất là phải đủ chỗ chứa cho mã nguồn (trước và sau khi xả nén), chỗ chứa để cài kernel và các *modules* mới sau khi biên dịch.

Đòi hỏi quan trọng khác là phải có một bộ công cụ cần thiết và đúng phiên bản. Không thể biên dịch được nhân nếu không thoả mãn yêu cầu này. Phiên bản cho bộ công cụ với mỗi phiên bản nhân khác nhau. Nên nhớ, nhóm phát triển nhân yêu cầu bạn phải có đúng phiên bản của các công cụ để đảm bảo việc biên dịch nhân thành công.

[2] extra-version
[3] development

## 4.1 Đòi hỏi cho nhân Linux 2.4.x

| Công cụ | Phiên bản tối thiểu | Cách xác định phiên bản |
|---|---|---|
| Gnu C | 2.91.66 | `gcc --version` |
| Gnu make | 3.77 | `make -version` |
| binutils | 2.9.1.0.25 | `ld -v` |
| util-linux | 2.10o | `fdformat --version` |
| modutils | 2.4.2 | `insmod -V` |
| e2fsprogs | 1.19 | `tune2fs` |
| reiserfsprogs | 3.x.0b | `reiserfsck 2>&1 \| grep reiserfsprogs` |
| pcmcia-cs | 3.1.21 | `cardmgr -V` |
| PPP | 2.4.0 | `pppd --version` |
| isdn4k-utils | 3.1pre1 | `isdnctrl 2>&1 \| grep version` |

Tham khảo thêm chi tiết các công cụ này và địa chỉ để tải các công cụ cho đúng (ít nhất phải cùng phiên bản đã cung cấp ở trên hoặc mới hơn) trong hồ sơ **Documentation/changes** của mã nguồn nhân mà bạn đang dự tính biên dịch.

## 4.2 Đòi hỏi cho nhân Linux 2.6.x

| Công cụ | Phiên bản tối thiểu | Cách xác định phiên bản |
|---|---|---|
| Gnu C | 2.95.3 | `gcc --version` |
| Gnu make | 3.78 | `make --version` |
| binutils | 2.12 | `ld -v` |
| util-linux | 2.10o | `fdformat --version` |
| module-init-tools | 0.9.10 | `depmod -V` |
| e2fsprogs | 1.29 | `tune2fs` |
| jfsutils | 1.1.3 | `fsck.jfs -V` |
| reiserfsprogs | 3.6.3 | `reiserfsck -V 2>&1\| grep reiserfsprogs` |
| xfsprogs | 2.1.0 | `xfs_db -V` |
| pcmcia-cs | 3.1.21 | `cardmgr -V` |
| quota-tools | 3.09 | `quota -V` |
| PPP | 2.4.0 | `pppd --version` |
| isdn4k-utils | 3.1pre1 | `isdnctrl 2>&1\| grep version` |
| nfs-utils | 1.0.5 | `showmount -version` |
| procps | 3.1.13 | `ps --version` |
| oprofile | 0.5.3 | `oprofiled --version` |

Tham khảo thêm chi tiết các công cụ này và địa chỉ để tải các công cụ cho đúng (ít nhất phải cùng phiên bản đã cung cấp ở trên hoặc mới hơn) trong hồ sơ **Documentation/changes** của mã nguồn nhân bạn đang dự tính biên dịch.

Nếu phiên bản của các công cụ trên máy cũ hơn các phiên bản đưa ra ở trên, bạn cần phải tải phiên bản mới (đã biên dịch) từ website của bản phân phối nào bạn đang dùng. Bạn cũng có thể chọn cách tải mã nguồn của từng công cụ về biên dịch lại. Cách này mất thời gian hơn rất nhiều và chỉ thích hợp cho những ai đã quen thuộc với vấn đề biên dịch mã nguồn trên Linux. Lợi điểm của cách này là bạn tạo cho mình một bộ công cụ rất "sạch" vì đã biên dịch theo ý, thích hợp với môi trường của máy (và vừa đủ).

Đối với phần cứng IA64, bạn cần phiên bản GCC cho 64bit, nên tham khảo chi tiết ở http://gcc.gnu.org/install/specific.html (nếu bạn may mắn có một con IA64 để thử)

# 5 Xác định cấu hình (hardware) của máy

Phần lớn người dùng bình thường ít khi quan tâm đến cấu hình của máy ngoại trừ có nhu cầu cụ thể. Ngay cả những ai dùng Linux đã lâu và không cần phải biên dịch lại nhân, cũng ít khi quan tâm đến cấu hình phần cứng của máy. Dù có biết nhiều hay ít về cấu hình phần cứng của máy mình dùng, bạn vẫn phải thu thập thông tin chính xác của cấu hình trước khi bắt tay vào việc điều chỉnh cấu hình cho việc biên dịch nhân.

Giả định mọi chỉnh lý đã ổn định và làm việc tốt đẹp (do trình cài đặt tự dò tìm (detect) trong khi cài Linux hay do bạn phải điều chỉnh lại), bạn cần biên dịch lại nhân Linux và cần thâu thập thông tin về cấu hình của máy, hai lệnh sau cung cấp các chi tiết phần cứng có trên máy:

```
# /sbin/lspci
```

Lệnh này liệt kê trọn bộ các "card" đang trực tiếp làm việc trên máy, phiên bản phần cứng và kiểu[4] của chúng.

```
# cat /proc/cpuinfo
```

để xem chi tiết bộ xử lý trung tâm (CPU) của máy là loại gì. Nếu bản Linux mà bạn dùng không có `/proc` filesystem thì bạn có thể dùng lệnh `dmesg` để thu thập thông tin về phần cứng trên máy của mình. Ngoài ra, lệnh `lsmod` cũng ít nhiều giúp bạn xác định các *modules* đang được dùng trên máy và tên của các *modules* này.

Những thông tin thâu thập được ở đây hết sức quan trọng trong giai đoạn điều chỉnh cấu hình cho việc biên dịch nhân. Nó giúp bạn xác định các chọn lựa đúng cho cấu hình máy, tránh đi những trở ngại có thể rất mất thời gian sau này.

# 6 Các bước chuẩn bị

## 6.1 Tạo một đĩa mềm khởi động cho nhân đang dùng

Đây là một bước cần thiết đề phòng sự cố gì đó khiến bạn không thể khởi động vào hệ thống Linux sau khi cài nhân mới. Trường hợp này hiếm khi xảy ra trong quá trình biên dịch nhân và cài nhân mới nếu bạn thực hiện và điều chỉnh đúng. Những sự cố với trình khởi động[5] ít thấy xảy ra vì trình khởi động đã được thiết lập hoàn chỉnh trước khi cập nhật nhân. Phần lớn giai đoạn điều chỉnh lại cấu hình của LILO hoặc GRUB không chính xác tạo trở ngại. Vấn đề này sẽ được đề cập sau. Trước mắt chúng ta cần tạo một đĩa mềm khởi động.

[4] model
[5] bootloader, LILO hoặc GRUB

Có rất nhiều cách để tạo một đĩa mềm khởi động lấy từ nhân hiện đang chạy trên máy. Thay vì trình bày nhiều cách khác nhau cho việc tạo đĩa mềm khởi động, ở đây tôi chỉ đề cập đến phương thức đơn giản nhất. Cách dễ nhất có lẽ là dùng `mkbootdisk`. Đây là một công cụ được cài mặc định trên các bản RedHat và hầu hết các bản phân phối dựa trên RedHat. Nếu công cụ này không hiện diện trên máy, bạn có thể tải mã nguồn về và biên dịch. Chạy lệnh:

```
# mkbootdisk --device /dev/fd0 `uname -r`
```

trong đó `` `uname -r` `` là lệnh để lấy phiên bản nhân hiện dùng trên máy. Nếu không muốn phiên bản này, bạn có thể gõ vào phiên bản nào đó theo ý (tất nhiên là phiên bản kernel này phải hiện hữu trên hệ thống).

`/dev/fd0` là "*device*" chỉ cho đĩa mềm thứ nhất trên máy (tương tự như drive A: trên DOS). Bạn phải chọn đúng "*device*" thì mới có dữ liệu viết vào đĩa mềm mình muốn tạo.

Quy trình này chỉ mất khoảng vài phút. Sau khi tạo đĩa mềm khởi động ở trên, bạn nên dùng nó để thử khởi động vào Linux trước khi thực hiện các bước kế tiếp. Nên nhớ phải chỉnh BIOS để cho phép máy khởi động từ A:

Ở trang http://www.yolinux.com/TUTORIALS/LinuxTutorialRecoveryAndBootDisk.html có các hướng dẫn tạo đĩa cấp cứu rất hay. Bạn nên tham khảo thêm nếu trên máy mình dùng không có sẵn công cụ `mkbootdisk`.

## 6.2 Tải mã nguồn

Mã nguồn của nhân Linux (cả phiên bản ổn định lẫn đang phát triển) có rất nhiều nơi trên Internet. Nên vào trang trung tâm của nhân Linux ở `http://www.kernel.org` và tham khảo danh sách "*mirrors*" để tìm nơi "gần" chỗ mình cư ngụ nhất để tải về. Nơi "gần" không nhất thiết là "gần" theo phương diện địa lý mà nên chọn "gần" nhất dựa trên "ping time". Chịu khó lấy vài địa chỉ trên mirror và ping những địa chỉ này để chọn lấy nơi có ping time ngắn nhất mà tải về. Bằng cách sẽ giúp bạn đỡ mất thời gian và tiện cho vấn đề chia xẻ băng thông.

Có nhiều cách tải mã nguồn. Bạn có thể dùng trình duyệt (browser) để tải qua giao thức http hoặc dùng một trình ftp nào đó để tải qua giao thức ftp. Bạn cũng có thể dùng `wget`. Có lẽ đây là cách tiện nhất và nhanh nhất nếu biết rõ địa chỉ và đường dẫn đến gói mình muốn tải. Ví dụ:

```
$ wget http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.26.tar.bz2
```

trong trường hợp này, gói cụ thể cần tải là `linux-2.4.26.tar.bz2`.

Mã nguồn ổn định của Linux kernel được nén ở hai dạng khác nhau: dạng có đuôi là `.gz` (dùng GNUzip để nén) và dạng có đuôi `.bz2` (dùng bzip2 để nén). Thông thường cả hai tiện ích nén/xả nén trên đều có sẵn trong các bản Linux thông dụng. Nếu không có sẵn

trên máy thì tìm trong trong CD của bản Linux (tham khảo thêm tài liệu của distribution mình dùng cho cách cài thêm phần mềm vào máy) hoặc tải về từ:

- `http://www.gzip.org` cho GNUzip
- `http://sources.redhat.com/bzip2/` cho bzip2

## 6.3 Kiểm tra thực tính của mã nguồn

Điều quan trọng khi tải mã nguồn của nhân, nên tải luôn chữ ký GPG .sign cho phiên bản tương ứng. Mục đích là để kiểm tra thực tính của mã nguồn được tải về. Khi mã nguồn của nhân Linux được công bố, chúng được dồn lại thành một gói (.tar) và sau đó được nén bằng GNUzip hoặc bzip2, cả hai loại này sau khi được nén đều được tạo "*chữ ký*" .sign.

Kiểm tra thực tính của mã nguồn được tải về bằng phương pháp kiểm tra "*chữ ký*" của từng gói mã nguồn là một thói quen cần thiết. Lý do: các mã nguồn mở nói chung được công bố và phổ biến rộng rãi, ai cũng có thể chỉnh sửa (một cách không chính thức và không được nhóm phát triển chính thức cho phép) rồi đưa lên một máy chủ nào đó trên Internet. Người dùng tải về, biên dịch và cài trên máy mà không kiểm tra thực tính của chúng (và mã nguồn này có những thay đổi mờ ám) thì hậu quả khó mà lường.

Quy trình kiểm tra "*chữ ký*" chỉ đơn giản gói gọn trong một dòng lệnh:

```
$ gpg --verify linux-2.4.26.tar.bz2.sign linux-2.4.26.tar.bz2
```

trong đó linux-2.4.26.tar.bz2.sign là "*chữ ký*" của gói linux-2.4.26.tar.bz2 được tải về từ server chứa mã nguồn nhân Linux linux-2.4.26.tar.bz2 là gói mã nguồn nhân Linux được nén bằng bzip2.

Trước khi có thể kiểm tra thành công bằng lệnh trên, bạn phải có `gpg` đã cài trong máy, tải và nhập chìa khóa công cộng (public key) của máy chủ chứa mã nguồn nhân Linux mà bạn tải về. Chi tiết hướng dẫn cho quy trình này ở http://www.kernel.org/signature.html

Quy trình tải mã nguồn nhân Linux và kiểm tra thực tính của mã nguồn này có thể tóm tắt bằng một ví dụ như sau:

Chuyển vào thư mục chứa mã nguồn của máy ở `/usr/src` là nơi thông thường. Đối với phiên bản nhân 2.6.x, bạn có thể dùng thư mục khác tùy ý:

```
$ cd /usr/src
```

Dùng wget để lấy một phiên bản mã nguồn từ server về ở dạng .bz2

```
$ wget http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.26.tar.bz2
```

dùng wget để lấy .sign của phiên bản mã nguồn vừa được tải về

```
$ wget http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.26.tar.bz2.sign
```

Dùng trình `gpg` với tùy chọn `-verify` để kiểm thực tính của mã nguồn vừa tải về

```
$ gpg --verify linux-2.6.10.tar.bz2.sign linux-2.6.10.tar.bz2
```

Ngoài phương pháp dùng chữ ký cho vấn đề kiểm chứng thực tính của mã nguồn (không chỉ mã nguồn của nhân Linux), bạn cũng thấy rất nhiều nơi trên Internet dùng "MD5sum" cho mục đích này (cho đến nay, mã nguồn nhân Linux kernel dùng chữ ký để kiểm chứng, không dùng MD5sum). Quy trình kiểm tra "MD5sum" chỉ đơn giản là một quy trình tạo một "MD5sum" từ mã nguồn được tải về trên máy và so sánh kết quả "MD5sum" này với hồ sơ "MD5sum" được tải về kèm với mã nguồn. Nếu "MD5sum" bạn tạo ra trên máy của mình với cùng gói mã nguồn mà không trùng hợp với "MD5sum" nguyên thuỷ tải về từ server thì thực tính của phần mã nguồn này không đáng tin cậy. Cách tốt nhất là chỉ nên tải mã nguồn ở những địa chỉ phố biến và đáng tin cậy. Cẩn thận hơn nữa (*really paranoid*), thì so sánh MD5sum với một số máy chủ chứa mã nguồn khác nhau.

Kiểm tra thực tính của mã nguồn bằng MD5 checksum khá đơn giản. Tiện ích `md5sum` có sẵn hầu như trên mọi bản phân phối. Lệnh tạo MD5 checksum đơn giản là lệnh:

```
# md5sum <file_cần_kiểm_tra>
```

sẽ tạo ra 1 chuỗi chữ và số tương tự như:

```
2fe2a5fabcc3a33722b4ffe05714bec3 *<file_cần_kiểm_tra>.
```

Nếu chuỗi này trùng với chuỗi được cung cấp chính thức với mã nguồn thì mã nguồn này có thực tính và đáng tin cậy.

## 6.4 Xả nén mã nguồn

Tùy vào gói mã nguồn được tải về thuộc dạng nén `.gz` hay `.bz2` mà dùng tiện ích thích hợp để xả nén. Như đã tóm tắt trong phần 6.3 ở trên, gói mã nguồn được chứa trong `/usr/src` (`wget` được chạy sau khi cd vào `/usr/src`), cho nên bạn phải ở trong thư mục này trước khi thao tác các bước kế tiếp (không thì các bước kế tiếp phải thêm và đường dẫn đến nơi chứa gói mã nguồn). Đối với phiên bản nhân 2.6.x, mã nguồn của nhân Linux có thể được xả, chứa và biên dịch từ bất cứ nơi đâu trên hệ thống. Tuy nhiên, để giữ cho hệ thống sạch và thống nhất, bạn nên giữ mã nguồn ở `/usr/src`.

Nếu gói mã nguồn có dạng .gz thì dùng:

```
$ gunzip linux-2.x.xx.tar.gz
```

x.xx là bất cứ phiên bản nào bạn tải về. Sau đó thực hiện tiếp:

```
$ tar xf linux-2.x.xx.tar
```

Lệnh này dùng tùy chọn `x` để xả (extract) và `f` để chỉ định file nào cần được xả, ở đây hồ sơ (file) cần được xả là `linux-2.x.xx.tar`.

Hai lệnh trên cũng có thể gộp chung lại như sau:

```
$ tar xfz linux-2.x.xx.tar.gz
```

lệnh này dùng thêm tùy chọn `z` để ngầm xả nén `.gz` file "*on-the-fly*" trước khi xả gói `tar`.

Hoặc có thể tạo cùng kết quả bằng cách khác nữa:

```
$ gzip -dc linux-2.x.xx.tar.gz | tar xvf -
```

cụm lệnh này dùng chương trình `gzip` để xả nén (tùy chọn -d) ra stdout (tùy chọn -c) và "tee" nó qua chương trình `tar` để xả gói tar ra "*on-the-fly*". Cả cách này và cách ở trên đều tiện dụng cho những ai eo hẹp dung lượng trên đĩa.

Nếu gói mã nguồn có dạng `.bz2` thì dùng:

```
$ bunzip2 linux-2.x.xx.tar.bz2
$ tar xf linux-2.x.xx.tar
```

Hai lệnh trên cũng có thể gộp chung lại như sau:

```
$ tar xfj linux-2.x.xx.tar.bz2
```

lệnh này dùng thêm tùy chọn `j` để ngầm xả nén `.bz2` file "*on-the-fly*" trước khi xả gói tar.

Hoặc có thể tạo cùng kết quả bằng cách khác nữa:

```
$ bzip2 -dc linux-2.x.xx.tar.bz2 | tar xvf -
```

cụm lệnh này dùng chương trình `bzip2` để xả nén (tùy chọn -d) ra stdout (tùy chọn -c) và "tee" nó qua chương trình `tar` để xả gói tar ra "*on-the-fly*". Cả cách này và cách ở trên đều tiện dụng cho những ai eo hẹp dung lượng trên đĩa.

Cả ba trường hợp đều cho kết quả là một thư mục có tên là `linux-2.x.xx` bên trong thư mục `/usr/src/`.

Trong phần này, chúng ta chỉ đề cập đến trường hợp tải trọn bộ mã nguồn của nhân Linux về để biên dịch. Trường hợp đã có mã nguồn cũ hơn của nhân Linux trên máy và chỉ cần tải bản vá lỗi và "vá" thì có quy trình khác. Vấn đề này sẽ đề cập sau.

## 6.5 Dùng "config" nào thì thích hợp?

Cấu hình biên dịch nhân Linux đơn giản là một "text file" chứa các biến với giá trị Y (Yes), N (No) hoặc M (Module). Các giá trị này được xử dụng trong quá trình biên dịch; chúng dùng để xác định những gì không được biên dịch, những gì được biên dịch và nếu được biên dịch thì sẽ theo dạng nào.

Tùy vào cách sắp xếp của mỗi bản phân phối Linux, cấu hình biên dịch nhân Linux nằm nhiều nơi khác nhau. Hồ sơ cấu hình theo mặc định của "vanilla" kernel nằm ở **./arch/i386/defconfig** (nếu dùng dòng phần cứng IA32 nói chung), các hồ sơ cấu hình khác cho những dòng phần cứng khác nằm ở **./arch/$ARCH/defconfig**; trong đó `$ARCH` là dòng phần cứng của máy. Nếu dùng cấu hình mặc định, không chỉnh sửa thì nhân sẽ được tái biên dịch trọn bộ theo giá trị mặc định và chắc hẳn, nhân này sẽ không thích hợp cho bạn (ngay cả nếu nó được biên dịch thành công). Điều này đi ngược lại mục đích cần biên dịch lại nhân Linux ngay từ đầu. Bạn có thể dùng hồ sơ cấu hình này để khởi đầu và chỉnh sửa giá trị cho thích hợp. Đây là một bước rất khó khăn cho những ai chưa từng đi qua giai đoạn này và không có sẵn một cấu hình biên dịch nhân hoàn chỉnh cho máy.

Cấu hình cho nhân hiện hữu trên máy cũng có thể nằm trong thư mục `/boot` ở dạng `config-2.x.xx` nếu bạn dùng nhân do RedHat (hoặc dựa trên RedHat) và một số bản phân phối khác cung cấp. Bạn có thể an toàn dùng cấu hình này và chỉnh sửa, loại bỏ các chi tiết (driver module) không cần dùng. Nếu hệ thống đã được biên dịch nhân trước đây, bạn có thể tìm thấy cấu hình biên dịch nhân Linux có tên là `.config`, được lưu trong thư mục `<KERNEL_SRC>` (nơi trước đây mã nguồn của nhân được xả nén và biên dịch).

# 7 Chỉnh cấu hình biên dịch nhân Linux

## 7.1 Thành phần của cấu hình biên dịch nhân Linux

Thành phần trong cấu hình biên dịch nhân Linux cho phiên bản 2.4.x và 2.6.x có một số điểm tương đồng và dị biệt. Tuy nhiên, quy trình chọn Y, N hoặc M cho các *modules* vẫn như nhau. Bước chọn lựa và chỉnh liệu cấu hình biên dịch nhân Linux là một bước mất nhiều thời gian nhất, nó cũng là một bước gây nhiều trở ngại nhất nếu chỉnh sửa không hợp lý hoặc thiếu sót.

### 7.1.1 Thành phần cấu hình biên dịch nhân Linux phiên bản 2.4.x

#### 7.1.1.1 Code Maturity Level Options

Chọn lựa của mục này cho phép dùng các *modules/drivers* còn ở trạng thái "alpha" (thử nghiệm). Nếu hệ thống làm việc là một máy production, cần tính ổn định cao thì nên tắt bỏ chọn lựa của phần này. Làm như thế sẽ tắt bỏ rất nhiều *modules/drivers* thuộc dạng

"alpha" trong những phần bên dưới. Nếu muốn thử dùng một số *modules/drivers* ở dạng *alpha* thì nên cho phép phần này (Y) và cẩn thận khi lựa chọn các *modules* được biên dịch sau này. Việc chọn lựa các "alpha" *drivers* ở chế độ mặc định của các nhân Linux trong nhiều bản phân phối Linux là một trong những nguyên nhân chính tạo nên tình trạng bất ổn định trên một số hệ thống Linux. Nếu chọn lựa các *driver* này một cách cẩn thận, cơ hội va phải tình trạng bất ổn định sẽ giảm thiểu rõ rệt.

#### 7.1.1.2 Loadable Module Support

Đây là chức năng nòng cốt của nhân Linux (*loadable module*). Như đã đề cập ở phần tổng quan (phần 1), các module có thể tải (loadable modules) là tiện dụng và linh động, cho nên bạn gần như sẽ chọn Y trong trường hợp này. Trong trường hợp bạn cần dùng module được viết thêm bên ngoài nhân chính thức (*3rd party modules*), bạn phải chọn "*enable set version information on all modules symbols*" trong mục này. Nếu bạn cần biên dịch trọn bộ các *drivers* thẳng vào nhân và không dùng modules (vì lý do bảo mật chẳng hạn), bạn có thể chọn N ở đây. Bạn cũng phải chọn "Y" cho trọn bộ các drivers trong cấu hình biên dịch nhân để thích hợp với chọn lựa "N" cho phần Loadable Module Support này.

#### 7.1.1.3 Processor Type and Features

Phần này có lẽ là phần tối quan trọng trong cấu hình biên dịch nhân Linux. Đây là nơi để chọn đúng CPU đang dùng trên máy. Ngoài ra còn rất nhiều chọn lựa khác nhau cho vấn đền system scheduling, SMP (symetrical multi-processing) nếu máy có nhiều CPU, hỗ trợ bộ nhớ lớn,.... Nếu bạn chọn CPU là i386 thì có lẽ sẽ không có sự cố vì i386 là architecture chung nhất (cả Intel và AMD CPU đều chạy với chọn lựa i386). Tuy nhiên, chọn lựa này sẽ không đạt hiệu năng tối đa và thích hợp cho từng loại CPU cụ thể. Nên chọn đúng CPU để bảo đảm hiệu năng của máy và nhất là để tránh trường hợp không thể khởi động vào Linux sau khi cài nhân mới (vì loại CPU chỉnh định cho nhân không đúng với CPU có trên máy hay nói một cách kỹ thuật, *instructions* giữa nhân và máy không đồng bộ).

#### 7.1.1.4 General Setup

Mục này cho phép chọn lựa các ứng dụng hỗ trợ cho những thiết bị (cards) trên máy như ISA, PCI, PCMCIA và các chức năng thuộc về vấn đề quản trị năng lượng cao cấp (Advanced Power Management).

#### 7.1.1.5 Memory Technology Devices

Phần này cho phép lựa chọn những ứng dụng thiết bị liên quan đến bộ nhớ. Nếu bạn dùng các thiết bị như máy ảnh số hoặc các loại *compact flash* thì bạn nên chỉnh lý phần này cho thích hợp.

#### 7.1.1.6 Block Devices

Đây là một phần rất quan trọng trong cấu hình biên dịch nhân Linux. Nó bao gồm các chọn lựa cho những thiết bị thông thường và cần thiết như đĩa cứng, đĩa mềm, băng lưu trữ cũng như các thiết bị điều tác (*controllers*) cho các cổng song song[6] và *RAID*. Hầu như các chọn lựa trong mục này đều cần thiết; đặc biệt là chức năng hỗ trợ *initrd* cần thiết để tải sẵn các *drivers* cần thiết ở dạng *module* trong quá trình khởi động máy.

#### 7.1.1.7 Multi-Device support (RAID and LVM)

Phần này chuyên chú đến các chức năng cần thiết cho hệ thống ở cấp độ máy chủ. Các chọn lựa ở đây hỗ trợ những thiết bị như RAID và LVM. Nếu máy của bạn hiện đang dùng RAID và LVM thì không thể bỏ qua phần này trong quá trình xác lập cấu hình biên dịch nhân Linux. Chọn lựa trong phần này đòi hỏi phải hiểu rõ nhu cầu dùng những công nghệ thuộc dạng này trên máy. Nếu máy không dùng đến những công nghệ này, bạn có thể an toàn tắt bỏ chúng (dùng N). Nên nhớ, nếu tắt bỏ RAID trong phần này thì phải tắt bỏ chọn lựa RAID trong phần "*block devices*" ở trên để tránh gặp phải lỗi biên dịch sau này.

#### 7.1.1.8 ATA/IDE/MFM/RLL support

Phần này bao gồm các chọn lựa và hỗ trợ cho IDE và ATAPI dùng trên các thiết bị tương thích với PC[7] (và trên nhiều *architecture* khác hiện có trên thị trường). Hầu hết các hệ thống cần các chức năng hỗ trợ trong phần này.

#### 7.1.1.9 Cryptography Support (CryptoAPI)

Đây là một phần khá mới và lý thú trong mã nguồn của nhân Linux 2.4.x (chỉ được giới thiệu và công bố trong các phiên bản sau này của 2.4.x). Phần này có những lựa chọn thuộc về vấn đề "mã hoá" cho filesystem. Bạn có thể biên dịch các chọn lựa trong mục này và xử dụng (hoặc không) trên máy tùy ý.

#### 7.1.1.10 Networking Options

Đây là một phần rất quan trọng trong cấu hình biên dịch nhân Linux nếu bạn muốn máy của mình kết nối với mạng. Nó bao gồm các chọn lựa cho cả hai chuẩn IPv4 và IPv6. Đây cũng là một phần hết sức phức tạp, cho nên, để có thể hiểu rõ và chọn lựa đúng cho hiệu năng tối đa của máy về mặt *networking*, bạn nên tham khảo các tài liệu về mạng Linux, ít nhất là nên đọc các tài liệu kèm theo trong mã nguồn nhân Linux ở <KERNL_SRC>/Documentation/networking/ (thường là /usr/src/linux-2.x.xx/Documentation/networking/).

---

[6] parallel ports
[7] pc-compatible

#### 7.1.1.11 SCSI Support

Phần chọn lựa cho SCSI ít được những người dùng bình thường quan tâm đến vì không mấy ai dùng SCSI cho máy con. Tuy nhiên nếu bạn dùng SCSI card (hoặc SCSI built-in trên bo mạch chủ (motherboard)) hoặc dùng CDR/W qua IDE nhưng chạy ở dạng mô phỏng SCSI thì phải điều chỉnh các chọn lựa trong mục này. Điều quan trọng cần nhớ, nếu không dùng tiện dụng initrd, khi chọn lựa SCSI cho một *filesystem* chạy trên đĩa SCSI bạn phải biên dịch trực tiếp các tùy chọn cho SCSI vào nhân thay vì dùng dưới dạng module. Nếu không, nhân sẽ treo trong giai đoạn khởi động vì module hỗ trợ SCSI chưa được tải lên trong giai đoạn này.

#### 7.1.1.12 Character Devices

Trong mục này có khá nhiều lựa chọn tập trung vào các thiết bị như nối tiếp[8] và song song[9], thiết bị chuột[10], joysticks (để chơi games). Tắt hoặc mở các lựa chọn trong mục này thường ít tạo ảnh hưởng nghiêm trọng.

#### 7.1.1.13 File Systems

Mục này chứa trọn bộ các chọn lựa liên quan đến hệ thống file (file system) và các loại *file system* được hỗ trợ trên Linux (bao gồm FAT, FAT32, NTFS, ISO cho CD-ROM....). Các *file system* phụ trợ như NTFS, FAT... có thể được biên dịch như một module cho nhân. Không nên biên dịch các modules cho *file system* dùng để "*mount*" trong giai đoạn khởi động như ext3, jbd mà nên biên dịch thẳng vào nhân (Lý do tương tự như đã đề cập trong phần "SCSI Support" ở trên). Cách này sẽ làm kích thước nhân lớn hơn nhưng sẽ an toàn và đơn giản hơn. Chức năng hỗ trợ `initrd` có thể dùng để tải các modules cần thiết trong quá trình khởi động nhân Linux nhưng phải nhớ bật chức năng này lên trong phần thiết bị dạng block[11]. Đây là vấn đề tùy chọn của từng cá nhân.

### 7.1.2 Thành phần cấu hình nhân Linux cho loạt nhân 2.6.x

#### 7.1.2.1 Code Maturity Level Options

Phần này tương tự như đã đề cập ở trên cho nhân 2.4.x.

#### 7.1.2.2 General Setup

Phần này tương tự như đã đề cập ở trên cho nhân 2.4.x.

[8] serial
[9] parellel
[10] mouse
[11] block devices

#### 7.1.2.3 Loadable Module Support

Phần này tương tự như đã đề cập ở trên cho nhân 2.4.x.

#### 7.1.2.4 Processor Type and Features

Phần này tương tự như đã đề cập ở trên cho nhân 2.4.x.

#### 7.1.2.5 Power Management Options

Phần này tương tự như đã đề cập ở trên cho nhân 2.4.x.

#### 7.1.2.6 Executable File Formats

Đây là một mục riêng biệt trong cấu hình biên dịch nhân của loạt nhân 2.6.x. Nếu bạn quan tâm đến "`a.out`", "`elf`" và "`misc`", nên nghiên cứu kỹ phần này qua các tài liệu kèm theo với mã nguồn nhân, đặt biệt cho các tiện dụng của "`misc`" ( <KERNEL_SRC>/Documentation/mono. <KERNEL_SRC>/Documentation/binfmt_misc.txt, <KERNEL_SRC>/Documentation/filesystem/proc.txt)

#### 7.1.2.7 Device Drivers

Đây là một mục mới trong phần cấu hình biên dịch nhân của loạt nhân 2.6.x. Thật ra *device drivers* nằm rải rác khắp nơi trong cấu hình biên dịch nhân của loạt nhân 2.4.x. Ở loạt nhân 2.6.x, mọi vấn đề liên quan đến "*device drivers*" được gom lại trong cùng một nhóm. Các chọn lựa thuộc về các thiết bị như card đồ họa[12], card âm thanh[13], USB, SCSI và vấn đề hiệu chỉnh chúng đều tập trung ở đây.

#### 7.1.2.8 File Systems

Phần này tương tự như đã đề cập ở trên cho nhân 2.4.x.

#### 7.1.2.9 Security Options

Phần này dành riêng cho các vấn đề về bảo mật của nhân. Cho đến nay vẫn còn đang phát triển, tuy nhiên, đây là phần đầy hứa hẹn cho một nhân Linux mang tính bảo mật cao.

## 7.2 Điều chỉnh cấu hình biên dịch nhân Linux

Sau đây là một số phương pháp để xác lập cấu hình biên dịch nhân Linux.

[12] graphic card
[13] sound card

### 7.2.1 Các công cụ để xác lập cấu hình

Như đã đề cập ở phần 6.5, mặc định cấu hình biên dịch nhân nằm ở `./arch/i386/defconfig`. Khi khởi động một công cụ (config tool) nó sẽ tự động đọc và dùng nội dung của file cấu hình mặc định này trước khi bạn chỉnh sửa.

Để chỉnh cấu hình biên dịch nhân Linux, chuyển vào thư mục chứa mã nguồn của nhân (đã xả nén):

```
$ cd /usr/src/linux-2.4.26
```

ví dụ này dùng nhân có số hiệu nhân 2.4.26 - xem lại phần xả nén ở phần 6.4) và việc đầu tiên rất nên làm đó là chạy lệnh:

```
$ make mrproper
```

Không kể bạn dùng bản phân phối Linux nào và phiên bản nhân Linux nào, bạn nên chạy lệnh này trước khi thực hiện quy trình biên dịch lại nhân. *Target* "mrproper" dùng để xoá hết tất cả những gì còn "*vất vưởng*" trong các thư mục chứa mã nguồn của nhân Linux để chắc chắn rằng mã nguồn trước khi được biên dịch phải ở tình trạng "*sạch sẽ*".

Có ba phương tiện "*config*" phổ biến có thể dùng để chỉnh cấu hình biên dịch nhân Linux. Sau khi chuyển vào thư mục `/usr/src/linux-2.4.26`, bạn có thể chọn một trong ba cách sau:

- make config
- make menuconfig
- make xconfig

Trong đó:

- **make config** là phương tiện đơn giản nhất và không đòi hỏi thêm bất cứ thư viện nào khác để chạy công cụ này. `make config` sẽ đưa ra một loạt câu hỏi và sau khi nhận được câu trả lời của bạn (Y, N, M như đã nói ở trên sau khi bạn nhấn phím Enter, xác nhận câu trả lời của mình), nó sẽ hình thành một cấu hình biên dịch nhân Linux. Nhược điểm của phương tiện này là ở chỗ, nếu bạn lỡ trả lời sai (chọn Y, N hoặc M và gõ phím Enter), bạn không thể quay ngược lại để điều chỉnh mà phải bắt đầu lại từ đầu. Phương tiện "`make config`" này chỉ tiện lợi cho những ai rất kinh nghiệm và nắm rõ mình cần gì trong cấu hình biên dịch nhân. Nó cũng tiện lợi cho quy trình chỉnh cấu hình biên dịch nhân từ xa (qua giao diện dòng lệnh (console) và không dùng được giao diện đồ họa vì lý do gì đó). Sau khi hoàn tất các câu hỏi, công cụ này sẽ lưu trữ một cấu hình biên dịch nhân (được lưu ở dạng `.config` trong thư mục chứa mã nguồn của nhân Linux) và sẵn sàng cho bước tạo các file phụ thuộc cho việc

biên dịch nhân. Nếu hồ sơ `.config` đã có từ lần biên dịch trước, nó sẽ bị viết chồng lên ở giai đoạn này.

- **make menuconfig** nâng cấp lên một mức cao hơn so với `make config`. Công cụ này cần thư viện và các binaries "`ncurses`" để tạo giao diện đồ họa (GUI) đơn giản. Với công cụ này, bạn có thể điều chỉnh lại các chi tiết tuỳ thích mà không phải bắt đầu lại từ đầu (nếu lỡ chọn sai) như dùng `make config`. Với giao diện đơn giản này, bạn có thể di chuyển, thay đổi các chọn lựa bằng cách dùng các phím mũi tên (lên xuống), chọn Y bằng phím Y, chọn N bằng phím N và chọn M bằng phím M. Với công cụ này, bạn cũng có thể tải một cấu hình biên dịch nhân có sẵn (đã làm từ trước và đã biên dịch thành công chẳng hạn) mà chẳng phải đi xuyên qua mọi chọn lựa để hình thành một cấu hình biên dịch nhân mới. Một đặc tính của công cụ này là nó chứa "trợ giúp ngữ cảnh"[14] (phần giúp đỡ hoặc thông tin cho từng mục trong quá trình điều chỉnh cấu hình). Nếu bạn không nắm rõ giá trị hoặc tác dụng của module nào đó, bộ phận trợ giúp này chắc chắn sẽ hữu ích. Sau khi đã hoàn thành các chọn lựa, bạn có thể lưu trữ một bản cấu hình trên máy để lần sau dùng lại. Ở giai đoạn này, một bản cấu hình được lưu lại (có tên `.config` trong thư mục chứa mã nguồn của Linux) và sẵn sàng cho bước tạo các file phụ thuộc cho quá trình biên dịch nhân. Tương tự như "`make config`", nếu hồ sơ `.config` đã tồn tại từ lần biên dịch trước, nó sẽ bị viết chồng lên.

- **make xconfig** có lẽ là phương tiện được dùng rộng rãi nhất, nhất là cho những hệ thống chạy X Window. **make xconfig** cần trọn bộ thư viện Qt và X Window để tạo các giao diện đồ họa[15]. Các chọn lựa và cách di chuyển trong giao diện này hoàn toàn giống như trường hợp dùng `menuconfig` và thêm một khả năng nữa là có thể dùng chuột[16] để chọn. Nếu bạn cần biên dịch lại nhân và có thể dùng X Window thì nên dùng công cụ `xconfig` này vì nó dễ dùng nhất.

Trên các phiên bản nhân Linux 2.6.x còn có thêm `make gconfig`. Tương tự như make `xconfig`, "`gconfig`" cũng tạo giao diện cấu hình đồ họa nhưng nó dựa trên thư viện Gtk.

### 7.2.2 Một số điểm cần chú ý trong giai đoạn hình thành cấu hình biên dịch nhân

- Dùng cấu hình biên dịch nhân có sẵn và điều chỉnh cho thích hợp với nhu cầu của mình. Nếu bạn dùng bản phân phối của RedHat, cấu hình biên dịch nhân có trong thư mục `/boot` ở dạng `config-<KERNEL_VERSION>` (hồ sơ cấu hình này từ các nhân do RedHat cung cấp). Để dùng cấu hình biên dịch nhân này, bạn có thể dùng `make menuconfig` hoặc `make xconfig` để tải cấu hình này lên và điều chỉnh theo

[14] help context sensitive
[15] GUI
[16] mouse

ý muốn. Thật ra không có bất cứ tài liệu nào có thể giải thích cặn kẽ từng chọn lựa cho mỗi cấu hình máy cả. Ở mức độ biên dịch lại một nhân, bạn cần hiểu cấu hình máy và những chọn lựa trong một cấu hình đến mức độ có thể quyết định chọn lựa những gì cho thích hợp. Bản phân phối Debian cũng lưu trữ các cấu hình biên dịch nhân trong thư mục `/boot` tương tự như bản phân phối RedHat. Các bản phân phối khác có một số điểm tương đồng và dị biệt, bạn nên tham khảo thêm các hướng dẫn cụ thể cho bản phân phối mình đang dùng.

- Lưu trữ cấu hình biên dịch nhân cho lần biên dịch kế tiếp. Nếu vì lý do gì đó khiến giai đoạn biên dịch nhân bị hỏng, bạn cần xem xét đoạn báo lỗi sau cùng khi trình dịch[17] thoát ra với "*tình trạng không thành công*" (*exit status is not 0*). Thông thường trình biên dịch thoát ra nửa chừng vì cách chọn lựa cấu hình biên dịch nhân có những điểm không thích hợp và thoả mãn (liên hệ nhau). Những lỗi được báo trên *console* giúp bạn xác định trục trặc nằm trong khu vực nào của cấu hình biên dịch nhân. Bạn chỉ cần tải hồ sơ biên dịch đã được lưu trữ lần cuối cùng lên và vào thẳng khu vực bị sự cố để xem xét và điều chỉnh, thay vì phải thiết lập từ đầu để tránh gặp những lỗi khác. Mỗi khi điều chỉnh và thay đổi cấu hình biên dịch nhân, bạn lại tiếp tục lưu lại một bản và đặt tên cho nó một cách hợp lý (ví dụ kernel-2.4.26-1 cho cấu hình thứ nhất, kernel-2.4.26-2 cho cấu hình thứ nhì.....).

- Nếu bạn dùng một nhân "*vanilla*", mã nguồn nhân được tải về từ http://sources.redhat.com/bzip2/kernel hoặc các máy chủ *mirror* (bài viết này tập trung chủ yếu vào nhân "*vanilla*"), sau khi hoàn thành bước `make config` hoặc `make menuconfig` hoặc `make xconfig`, cấu hình biên dịch nhân đã được bạn điều chỉnh và chọn lựa sẽ được lưu trữ ở dạng ẩn[18] ở `./.config` (các hồ sơ và thư mục bắt đầu bằng dấu chấm . được xếp loại là file ẩn, chỉ có `ls -la` mới thấy chúng). Lần kế tiếp, sau khi chạy một trong số lệnh trên và tiếp nhận các thay đổi thì cấu hình biên dịch nhân lại tự động viết chồng lên `./.config`. Cho nên, bạn có thể lưu trữ hồ sơ `./.config` thành một hồ sơ có tên khác trước khi điều chỉnh và chọn lựa lần kế tiếp.

- Ngoài các lệnh `make config`, `make menuconfig`, `make xconfig` hoặc `make gconfig` (cho loạt nhân 2.6.x), bạn còn có thể dùng một lệnh khác cho cả kernel 2.4.x và 2.6.x là: `make oldconfig`. Lệnh này là một trường hợp đặc biệt dùng để đọc và dùng các tùy chọn đã có sẵn trong `./.config` mà không cho bạn cơ hội để điều chỉnh và chọn lựa. Trường hợp này rất tiện lợi nếu bạn đã điều chỉnh và chọn lựa thành công một cấu hình biên dịch nhân cho mình.

Phiên bản nhân 2.6.x còn có bốn *target* cho bước này:

[17] compiler
[18] hidden

- **make defconfig**: tạo một cấu hình biên dịch nhân mới với chế độ mặc định cho tất cả chọn lựa
- **make allmodconfig**: tạo một cấu hình biên dịch nhân mới với chế độ chọn lựa các modules khi có thể được
- **make allyesconfig**: tạo một cấu hình biên dịch nhân mới với chế độ tiếp nhận yes (Y) cho tất cả chọn lựa
- **make allnoconfig**: tạo một cấu hình biên dịch nhân mới với chế độ tiếp nhận **N** cho tất cả chọn lựa. Chế độ này sẽ tạo ra một nhân rất nhỏ và đơn giản.

Hiếm khi bạn điều chỉnh một cấu hình biên dịch nhân Linux lần đầu mà không hề bị lỗi trong khi biên dịch. Cách lưu trữ từng cấu hình cho mỗi lần hiệu chỉnh là cách tốt nhất để bảo đảm "*lỗi*" lần trước sẽ không tái diễn. Nếu bạn chỉ đơn giản dùng cấu hình biên dịch nhân có sẵn (như RedHat Linux chứa trong `/boot`) và không điều chỉnh gì cả thì ngoài mục đích vá lỗi, lối dùng này chẳng có tác dụng gì về mặt nâng cao hiệu năng của máy.

# 8 Các bước biên dịch

## 8.1 Bước tạo *dependency*, dọn dẹp và tạo nhân

Bước này có thể chạy ba lệnh:

```
# make dep
# make clean
# make bzImage
```

hoặc gom chung lại thành một nhóm:

```
# make dep clean bzImage
```

Nếu bạn chạy `dep`, `clean` và `bzImage` riêng biệt thì phải trông chừng khi nào lệnh thứ nhất hoàn tất để tiếp tục chạy lệnh thứ nhì và tiếp theo. Nếu bạn chạy ba lệnh một lượt thì lệnh thứ nhì tự động nối tiếp lệnh thứ nhất và lệnh thứ ba nối tiếp lệnh thứ nhì. Bạn không cần phải chờ đợi.

- bước "dep" là bước tạo *dependencies* và các file bao gồm[19] cần thiết cho việc biên dịch nhân. Bước này có thể mất nhiều phút, tùy vào CPU của từng máy. Đối với loạt nhân 2.6.x, bước này không cần thiết nữa.

- sau khi xong bước "dep", bước "clean" dùng để dọn dẹp tất cả những *objects* vụn vặt, không còn cần thiết vì quá trình tạo file phụ thuộc đã hoàn tất ở trên.

- bước kế tiếp "bzImage" là bước tạo nhân. Đây là bước hết sức quan trọng trong ba bước. Nếu có sự cố gì xảy ra thì phải quay lại điều chỉnh cấu hình biên dịch nhân và thực hiện lại các bước "make dep", "make clean" trở lại (cần phải chạy một số lệnh dọn dẹp trước khi make dep clean bzImage trở lại, vấn đề này sẽ được đề cập trong phần 10). Trên máy chạy Athlon Thunderbird 1.4Ghz, tôi mất chừng 10 phút để hoàn thành bước này. Trên một máy Pentium 233MMX cũ, tôi mất hơn 40 phút mới hoàn thành bước tạo *kernel image*.

Nếu ba bước trên hoàn toàn thành công, bạn có thể tìm thấy nhân[20] nằm trong thư mục ./arch/$ARCH/boot, trong đó $ARCH là dòng phần cứng của nhân bạn muốn biên dịch. Nếu máy bạn thuộc dạng i386, bạn sẽ tìm thấy nhân trong ./arch/i386/boot. Nhân này đã được tạo ra nhưng chưa được cài ở bước này, nó chỉ lưu trong thư mục trên cho các bước về sau.

Thật ra có thể tạo nhiều dạng "kernel image". Dạng *kernel image* được tạo từ "make bzImage" là dạng phổ biến nhất hiện nay vì nó nén *kernel image* tốt nhất và thích hợp với hầu hết các loại máy.

Bạn cũng có thể dùng:

```
# make zImage
```

make zDisk hoặc make zLilo để tạo kernel image nếu kernel dự kiến rất nhỏ và không cần kỹ thuật nén cao độ như "bz". Dùng các dạng này cũng thích hợp trong trường hợp máy của bạn quá cũ và có thể có sự cố với "bzImage". Chỉ cần nắm một cách khái quát như sau:

- Phần bz hoặc z đi trước các image ở trên chỉ định cho loại nén nào được dùng với kernel image.

- Phần Image hoặc Disk hoặc Lilo chỉ định cho "loại" kernel image.

- Kernel image này được xả nén "on-the-fly" trong quá trình boot vào Linux sau này.

Xuyên qua ba bước ở trên, bạn sẽ thấy vô số thông điệp chạy trên console (ở kernel 2.6.x thông điệp chạy trên console ít hơn rất nhiều). Bất cứ lỗi nào (error) được báo trong bước

[19] includes
[20] kernel image

này đều phải điều chỉnh cấu hình biên dịch nhân và trở lại bước "make dep". Cho đến giai đoạn này, lý do gây ra lỗi thường là:

- đồ nghề dùng để biên dịch không đúng phiên bản (xem phần 4.1 và 4.2 cho loạt nhân này bạn đang biên dịch)
- điều chỉnh sai hoặc thiếu một số chọn lựa nào đó trong cấu hình biên dịch nhân. Xem lỗi báo trước khi compiler thoát ra để xác định lỗi này thuộc phần nào trong cấu hình biên dịch nhân mà chỉnh lại cho thích hợp, nên dùng phương pháp tải và lưu trữ cấu hình biên dịch nhân đã đề cập trong phần 7.2.2 ở trên.
- cấu hình máy quá thấp (memory / diskspace) không đủ để thực hiện ba bước ở trên. Nếu gặp sự cố này, nên nâng cấp máy hoặc dùng một máy khác để build kernel cho máy này.

Loạt nhân 2.6.x đơn giản hoá chỉ với một target "make all". Target này bao gồm luôn phần "make modules" trong bước 8.2 kế tiếp.

## 8.2 Bước tạo modules và cài modules

Bước này có thể chạy hai lệnh:

```
make modules
make modules_install
```

hoặc gom chung lại thành một dòng:

```
make modules modules_install
```

Điểm khác biệt giữa cách chạy hai lệnh riêng biệt hoặc chạy chung một dòng lệnh ở đây nằm ở chỗ:

- bạn có thể chỉ muốn biên dịch modules cho kernel mà không muốn cài (install) trên máy ngay sau khi các modules được biên dịch xong,
- hoặc bạn chỉ muốn biên dịch modules trên máy này rồi sẽ mang qua máy khác để cài.

Thông thường "`make modules modules_install`" đi chung vì ít người build modules trên một máy rồi mang đi cài trên một máy khác. Nếu chạy hai lệnh này một lượt, bạn phải chạy ở chế độ "super user" không thì modules không install được vì chỉ có root (super user) mới có thể "install" các modules vừa được biên dịch. Nếu bạn tách rời hai lệnh trên thì các lệnh tách rời như sau:

```
$ make modules
```

chạy bằng user account bình thường

```
# su
[enter password]
```

chuyển sang chế độ "super user"

```
make modules_install
```

cài modules vừa biên dịch xong.

Bước "`make modules`" là bước biên dịch và tạo ra các modules (mà bạn đã chọn ở dạng M trong quá trình chỉnh lý cấu hình biên dịch nhân). Các modules đã được biên dịch sẽ được lưu trữ trong các thư mục thích ứng với từng nhóm "drivers" trong cây mã nguồn (*kernel source tree*). Giai đoạn này là giai đoạn biên dịch lâu nhất trong trọn bộ quá trình compiler thực sự biên dịch mã nguồn của kernel. Trên một máy chạy Athlon Thunderbird 1.4Ghz, bước này mất chừng 25 phút. Trong khi đó cùng số lượng modules cần biên dịch chạy trên máy Pentium 233MMX mất chừng trên 4 giờ đồng hồ.

Bước "`make modules_install`" sẽ "cài" các modules vừa được biên dịch vào thư mục `/lib/modules/<kernel_version>`. Nếu liệt kê thư mục này (ls), bạn sẽ thấy ít nhất một thư mục chứa modules cho kernel đang chạy trên máy hoặc nhiều thư mục cho nhiều phiên bản kernel trước đây (có từ quy trình cập nhật kernel bằng rpm hoặc quy trình nào đó tuỳ theo bản phân phối, hoặc từ quy trình biên dịch kernel tương tự như bài viết này). Khi boot Linux bằng một phiên bản kernel nào đó có trên máy, các modules thuộc kernel này (trong thư mục thích ứng với kernel version) sẽ được ứng tải.

Đối với loạt nhân 2.4.x, bạn có thể tham khảo chi tiết thông tin về modules, cách biên dịch modules tổng quát và cách xử dụng modules (thuộc user space) trong hồ sơ `./Documentation/modules.txt` thuộc mã nguồn kernel bạn dự định biên dịch.

Đối với loạt nhân 2.6.x, bạn có thể tham khảo chi tiết thông tin về modules, cách biên dịch modules tổng quát và cách xử dụng modules (thuộc user space) trong ba hồ sơ ./Documentation/kbuild/modules.txt, ./Documentation/networking/net-modules.txt và ./Documentation/sound/oss/README.modules thuộc mã nguồn kernel bạn dự định biên dịch. Riêng với loạt nhân 2.6.x, bước "make modules" có thể thực hiện từ "make all" và bước "make modules_install" chỉ thực hiện riêng (ở chế độ super user) để cài các modules đã được biên dịch.

## 8.3 Tách rời mã nguồn và hồ sơ output trên loạt nhân 2.6.x

Nếu bạn đang dùng loạt nhân 2.4.x thì không cần tham khảo thông tin của mục này. Những thông tin trong mục này chỉ giới thiệu thêm một số tiện ích hữu dụng cho quy trình chuẩn bị và biên dịch nhân 2.6.x.

### 8.3.1 "make help", một tiện ích mới trên loạt nhân 2.6.x

Ngoài những điểm khác biệt trong các *make target* đã được đề cập ở phần 8.1 và 8.2, trên loạt nhân 2.6.x, bạn có thể xử dụng một tiện ích khá hay mà kernel 2.4.x không có đó là phần "help" trước khi "make" mã nguồn của nhân Linux. Tất nhiên bạn phải chạy lệnh này sau khi vào trong thư mục chứa mã nguồn nhân Linux:

```
$ cd /usr/src/linux-2.6.6
```

dùng kernel 2.6.6 trong trường hợp này

```
$ make help
```

sẽ cho thông tin trợ giúp như sau:

```
bash-2.05b$ make help
Cleaning targets:
  clean        - remove most generated files but keep the config
  mrproper     - remove all generated files + config + various backup files

Configuration targets:
  oldconfig    - Update current config utilising a line-oriented program
  menuconfig   - Update current config utilising a menu based program
  xconfig      - Update current config utilising a QT based front-end
  gconfig      - Update current config utilising a GTK based front-end
  defconfig    - New config with default answer to all options
  allmodconfig - New config selecting modules when possible
  allyesconfig - New config where all options are accepted with yes
  allnoconfig  - New minimal config

Other generic targets:
  all             - Build all targets marked with [*]
* vmlinux         - Build the bare kernel
* modules         - Build all modules
  modules_install - Install all modules
  dir/            - Build all files in dir and below
  dir/file.[ois]  - Build specified target only
  rpm             - Build a kernel as an RPM package
  tags/TAGS       - Generate tags file for editors
  cscope          - Generate cscope index

Documentation targets:
  Linux kernel internal documentation in different formats:
  sgmldocs (SGML), psdocs (Postscript), pdfdocs (PDF)
  htmldocs (HTML), mandocs (man pages, use installmandocs to install)

Architecture specific targets (i386):
* bzImage       - Compressed kernel image (arch/i386/boot/bzImage)
  install       - Install kernel using
                  (your) ~/bin/installkernel or
                  (distribution) /sbin/installkernel or
                  install to $(INSTALL_PATH) and run lilo
  bzdisk        - Create a boot floppy in /dev/fd0
  fdimage       - Create a boot floppy image
```

```
  make V=0|1 [targets] 0 => quiet build (default), 1 => verbose build
  make O=dir [targets] Locate all output files in "dir", including .config
  make C=1   [targets] Check all c source with checker tool

Execute "make" or "make all" to build all targets marked with [*]
For further info see the ./README file
bash-2.05b$
```

Thông tin trên cho thấy "Makefile" chính của loạt nhân 2.6.x bao gồm các mục tiêu (target) biên dịch khi chạy make help. Với thông tin này, bạn có thể chọn các target make theo ý muốn mà không phải kiểm tra trong "Makefile" như với loạt nhân 2.4.x (loạt nhân 2.4.x không có "`make help`" như loạt nhân 2.6.x và loạt nhân 2.4.x không có nhiều make targets như loạt nhân 2.6.x). Điểm đặt biệt cần quan tâm là ba chọn lựa cuối trong thông tin "`make help`" cung cấp:

```
make V=0|1 [targets] 0 => quiet build (default), 1 => verbose build
  make O=dir [targets] Locate all output files in "dir", including .config
  make C=1   [targets] Check all c source with checker tool
```

Một trong những chọn lựa quan trọng ở đây là nó cho phép bạn lưu trữ trọn bộ các hồ sơ output trong quá trình biên dịch vào một thư mục riêng biệt thay vì chứa chung với mã nguồn của kernel.

### 8.3.2 Tách rời mã nguồn và output files

Loạt nhân 2.6.x cho phép bạn tách rời mã nguồn của kernel và các hồ sơ output được tạo trong quá trình compile, các hồ sơ ẩn[21] như .config, .depend.... trong các bước đề cập ở phần 7 và ?? cũng sẽ được lưu trữ ở thư mục nào bạn muốn dùng cho output files. Với phương tiện này, mã nguồn và các hồ sơ output sẽ không xen kẽ chung. Điểm quan trọng cần nhớ là khi đã dùng chọn lựa này thì phải dùng cho các bước "make" khác trong suốt quá trình biên dịch. Ví dụ, bạn có thể khởi đầu bằng:

```
# make O=/path/to/output xconfig
```

thì các bước kế tiếp sẽ là:

```
# make O=/path/to/output all
# make O=/path/to/output modules_install
```

target "all" bao gồm "dep, clean, bzImage, modules". Chạy lệnh này bằng super user để cài modules của kernel.

# 9 Cài đặt nhân

Phần này giới thiệu hai cách cài nhân vừa biên dịch và chỉnh định *boot loader*.

[21] hidden

## 9.1 Cài đặt với "`make install`"

Ít người dùng đến chức năng "`make install`" này vì một số bản phân phối không có các tiện ích cần thiết để thực hiện trọn vẹn bước này. "`make install`" tiện lợi và an toàn hơn cài bằng tay vì nó thao tác các bước cần thiết để thiết lập nhân mới trên hệ thống. Các bước này bao gồm quy trình lưu trữ nhân cũ (trong thư mục `/boot`), copy nhân mới, copy `System.map` mới, điều chỉnh *boot loader configuration* (`lilo.conf` hoặc `grub.conf`) và cập nhật *boot loader*.

Bước "`make install`" dựa trên hồ sơ `Makefile` và `install.sh`, một shell script thuộc thư mục `./arch/$ARCH/boot`. Shell script `install.sh` "gọi" một số shell script khác như `/sbin/installkernel` và `/sbin/new-kernel-pkg`, ngoài ra các *shell scripts* này còn dựa vào một *binary* có tên là "`grubby`" để tạo thông tin trong `grub.conf` nếu bạn dùng GRUB. Các shell scripts "`installkernel`" và "`new-kernel-install`" thuộc gói `mkinitrd` của RedHat, các bản phân phối khác có những ứng dụng tương tự. Nếu bản phân phối bạn dùng không có gói tương tự, bạn phải cài kernel bằng tay (phần 9.2) hoặc tạo các script tương tự để thực hiện bước này. Trong khuôn khổ giới hạn của bài viết, tôi không đi sâu vào vấn đề tạo các script tiện ích.

Để cài nhân Linux mới, bạn chỉ đơn giản chạy lệnh `make install` ở chế độ *super user* từ trong thư mục chứa mã nguồn của nhân Linux kernel. Sau khi hoàn tất bước "`make install`" bạn nên kiểm tra lại cấu hình của trình khởi động[22] trên máy và chạy các lệnh tương ứng (nếu cần) để chỉnh định trình khởi động cho chính xác.

### 9.1.1 Đối với GRUB

Ví dụ bạn có hai phiên bản nhân trên máy 2.4.20 (phiên bản đang chạy) và 2.4.26 (phiên bản vừa được biên dịch). Sau khi chạy "`make install`", `grub.conf` có nội dung như sau:

```
default=1
timeout=20
splashimage=(hd0,0)/boot/grub/splash.xpm.gz

title Linux (2.4.26)
    root (hd0,0)
    kernel /boot/vmlinuz-2.4.26 ro root=/dev/hda1
    initrd=/boot/initrd-2.4.26.img

title Linux (2.4.20)
    root (hd0,0)
    kernel /boot/vmlinuz-2.4.20 ro root=/dev/hda1
    initrd=/boot/initrd-2.4.20.img
```

[22] boot loader

- Chi tiết cần chú ý là biến `default`. Trong ví dụ trên, bạn có hai nhân trong cấu hình GRUB cho các phiên bản 2.4.26 và 2.4.20. Nếu bạn muốn khởi động nhân 2.4.26 theo mặc định thì giá trị của `default` phải là `0` (grub đếm thứ tự các nhân từ 0). Khi chạy "`make install`", các tiện ích của "`install`" tự động đưa vào các chi tiết thuộc kernel mới vào cấu hình GRUB. Tuy nhiên, giá trị default vẫn giữ ở giá trị chỉ định cho nhân hiện đang hoạt động trên máy. Bạn cần chỉnh giá trị này để buộc trình khởi động tải lên phiên bản nhân mới. Một chi tiết hết sức quan trọng bạn cần chú ý là giá trị `root (hdX,Y)`. Nếu GRUB đã được cài trong lúc cài đặt hệ thống từ CD và đã hoạt động hoàn chỉnh, bạn không nên thay đổi giá trị này. Giá trị này chỉ cần thay đổi nếu bạn thêm đĩa cứng và thay đổi các phân vùng[23] trên máy.

- sau khi chỉnh định và lưu trữ `grub.conf` thích hợp, bạn chỉ cần khởi động lại máy. Nếu bạn dùng GRUB làm trình khởi động thì công tác biên dịch lại nhân Linux hoàn thành ở đây.

- giải pháp phòng bị: trường hợp không thể boot vào nhân mới rất đơn giản nếu dùng GRUB làm trình khởi động. Bạn chỉ cần thêm một dòng `fallback 1` vào cấu hình `grub.conf` là đủ. Tùy chọn này cho GRUB biết nếu dùng "`default=0`" để khởi động nhân mới nhất (2.4.26 trong ví dụ này) nhưng không thành công vì lý do nào đó thì thử khởi động lại với nhân cũ hơn (2.4.20). Xem thêm ở phần 10 nếu không thể khởi động được vào Linux vì trình khởi động bị hỏng.

### 9.1.2 Đối với LILO

Ví dụ bạn có hai phiên bản nhân trên máy 2.4.20 (phiên bản đang chạy) và 2.4.26 (phiên bản vừa được biên dịch), sau khi chạy "`make install`", `lilo.conf` có nội dung như sau:

```
prompt
timeout=50
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message

image=/boot/vmlinuz-2.4.20
initrd=/boot/initrd-2.4.20.img
root=/dev/hda1
label=linux-2.4.20
read-only
```

[23] partitions

```
image=/boot/vmlinuz
initrd=/boot/initrd-2.4.26.img
root=/dev/hda1
label=linux
read-only
```

- chi tiết cần chú ý là biến `default`. Trong ví dụ trên, bạn có hai nhân trong cấu hình LILO cho các phiên bản 2.4.26 và 2.4.20. Nếu bạn muốn khởi động nhân 2.4.26 thì giá trị của `default` phải là giá trị `label` (nhãn) thuộc nhân nào bạn muốn dùng. Trong trường hợp này, nhãn (label) có giá trị là `linux` chỉ định cho nhân 2.4.26. Khi chạy "`make install`", các tiện ích của "`install`" đưa vào các chi tiết của nhân mới vào cấu hình LILO. Ở đây `vmlinuz` là liên kết biểu tượng[24] của hồ sơ[25] `vmlinuz-2.4.26`. Bạn nên kiểm tra lại giá trị `default` để bảo đảm trình khởi động sẽ tải nhân vừa biên dịch khi khởi động.

- sau khi chỉnh định và lưu trữ `lilo.conf` theo ý muốn, bạn phải chạy lệnh:

```
# /sbin/lilo
```

  và chú ý trường hợp hệ thống báo lỗi trong bước cài lilo làm trình khởi động. Nếu có, điều chỉnh cho chính xác và thực hiện lại lệnh trên. Với ví dụ trên, bạn sẽ thấy hiển thị kết quả như sau:

```
Added linux*
Added linux-2.4.20
```

  Nhân nào đi kèm với dấu hoa thị (*) là nhân sẽ được khởi động theo mặc định.

- với LILO, giải pháp đơn giản nhất đề phòng trường hợp không thể khởi động vào nhân mới mang tính tạm thời. Trước khi khởi động lại máy dùng lệnh sau:

```
# /sbin/lilo -R linux
```

  rồi chạy

```
# reboot
```

- Nên nhớ các lệnh trên được đưa ra như một ví dụ cho nhân 2.4.20 và 2.4.26 với cấu hình khởi động như trên. Bạn phải điều chỉnh đúng phiên bản nhân mà bạn đang biên dịch.

- Lệnh thứ nhất cho LILO biết lần kế tiếp máy khởi động lại thì thử dùng nhân 2.4.26. Nếu không thành công thì không lấy nhân 2.4.26 làm nhân mặc định và lần boot kế

[24] symbolic link
[25] file

tiếp sẽ dùng nhân 2.4.20 (nhân này chắc chắn phải làm việc được vì nó đã dùng để biên dịch nhân 2.4.26).

- Lệnh thứ nhì chỉ đơn giản ra lệnh cho máy khởi động lại.
- Nếu dùng lệnh "`/sbin/lilo -R linux-x.xx.xx`" và khởi động vào nhân mới thành công thì bạn cần chỉ định cho nhân x.xx.xx làm nhân mặc định rồi mới chạy "`/sbin/lilo -v`" như đã nói ở trên (trong phần biến "`default`" của `lilo.conf`).

## 9.2 Các bước cài đặt bằng tay

Các bước cài đặt "bằng tay" tương tự như các bước "`make install`" ở trên nhưng được thao tác "bằng tay". Thật ra quy trình này rất đơn giản, điều bạn cần lưu ý là phải thực hiện chính xác để tránh những trở ngại trong bước này và trong giai đoạn khởi động vào nhân mới.

### 9.2.1 Tạo initrd

Trường hợp bạn biên dịch các *drivers* quan trọng ở dạng *modules* có liên hệ đến quy trình khởi động của Linux (như SCSI driver, RAID driver, các loại *filesystem* mà root filesystem dùng như ext3, jbd...) thì chắc chắn bạn phải cần đến `initrd`[26]. Mục đích chính của `initrd` là tải sẵn các *driver* cần thiết cho nhân trong quá trình khởi động. Nếu không muốn dùng `initrd`, bạn phải biên dịch các *driver* trực tiếp vào nhân[27]. Nên chú ý một số bản phân phối Linux không dùng `initrd`. Họ khuyến khích biên dịch các *driver* liên hệ đến quy trình khởi động trực tiếp vào nhân. Muốn tham khảo thêm chi tiết về *RAM disk* cho trường hợp này, xem <KERNEL_SRC>/Documentation/ramdisk.txt.

Quy trình tạo `initrd` rất đơn giản, chỉ cần chạy lệnh:

```
# /sbin/mkinitrd /boot/initrd-<KERNEL_VERSION>.img <KERNEL_VERSION>
```

trong đó:

- Tham số thứ nhất `/boot/initrd-<KERNEL_VERSION>.img` chỉ định cho hồ sơ và thư mục chứa hồ sơ `initrd`. Thông thường `initrd` của nhân được chứa trong thư mục `/boot` cùng với các thông tin và hồ sơ khác cần thiết cho quy trình khởi động.
- Tham số `<KERNEL_VERSION>` thứ nhì chính là nhân nào bạn muốn tạo `initrd` cho nó. Tất nhiên thư mục chứa các modules cho phiên bản nhân này phải có trong `/lib/modules/`, nếu không bạn được system báo có lỗi.

[26] INITial Ram Disk
[27] hay còn gọi là static compile

Tùy thuộc vào bản phân phối, `mkinitrd` đòi hỏi thêm các thông số cụ thể để chỉ đường dẫn đến nhân. Nếu gặp trở ngại trong bước tạo `mkinitrd` bạn nên tham khảo tài liệu cụ thể cho bản phân phối mình đang dùng hoặc tối thiểu là xem `man mkinitrd` và tài liệu <KERNEL_SRC>/Documentation/initrd.txt để xem thêm các thông tin cần thiết.

Một điểm đáng chú ý là từ loạt nhân 2.5.x[28] trở đi, `initramfs` được phát triển với mục đích hỗ trợ và sẽ đi đến chỗ thay thế `initrd`. Ưu điểm nổi bật của `initramfs` là nó có thể chứa các bộ lưu trữ ở dạng cpio "newc" hoặc "crc" (được nén hoặc không được nén). `initramfs` cho đến nay chưa phổ biến và ứng dụng rộng rãi trên các bản phân phối Linux. Tuy nhiên, hướng phát triển và ứng dụng `initramfs` có vẻ đầy hứa hẹn.

### 9.2.2 Copy nhân và System.map

Sau khi hoàn thành bước "`make modules_install`" (phần 8.3), lúc này bạn đã có trọn bộ các bộ phận cần thiết cho nhân mới bao gồm cả kernel image và các modules thuộc nhân này.

- copy `bzImage` từ `<KERNEL_SRC>/arch/i386/boot/` đến thư mục `/boot`, ví dụ:

```
# cp /usr/src/linux-2.4.26/arch/i386/boot/bzImage /boot/bzImage-
2.4.26
```

- Trình cài đặt của RedHat và một số bản phân phối khác bao gồm bước copy `bzImage` thành `vmlinuz`, bạn có thể thực hiện (hay không tùy ý, bước này tương tự như bước ở trên) như sau:

```
# cp /usr/src/linux-2.4.26/arch/i386/boot/bzImage /boot/vmlinuz-
2.4.26
```

- kế tiếp là xoá liên kết[29] cũ (nếu có) của `vmlinuz` trong thư mục `/boot`:

```
# rm -f /boot/vmlinuz
```

- và sau đó tạo liên kết mới cho `vmlinuz-2.4.26` thành:

```
# ln -s /boot/vmlinuz-2.4.26 /boot/vmlinuz
```

- Tất nhiên bạn phải điều chỉnh lại *boot loader* để thích ứng với cách gọi "`bzImage`" hoặc "`vmlinuz`" này cho giá trị image (trong lilo.conf) hoặc giá trị kernel (trong grub.conf). Cách dùng và cách gọi `bzImage` và `vmlinuz` tạo khá nhiều bối rối cho người dùng Linux khi tiếp cận quy trình biên dịch nhân. Một số bản phân phối Linux dùng `bzImage`, một số khác lại dùng `vmlinuz`. Dù gì đi chăng nữa, đây cũng chỉ

[28] development kernel
[29] symbolic link

là cách dùng và cách gọi; bạn nên dùng theo cách bản phân phối Linux nào có trên máy.

- phần còn lại là bước copy hồ sơ `System.map`:

```
# cp /usr/src/linux-2.4.26/System.map-2.4.26 /boot/System.map-
2.4.26
```

- kế tiếp là xoá liên kết cũ của `System.map` trong thư mục `/boot`:

```
# rm -f /boot/System.map
```

- và sau đó, tạo liên kết mới cho `System.map`:

```
# ln -s /boot/System.map-2.4.26 /boot/System.map
```

### 9.2.3 Chỉnh cấu hình của bootloader config

#### 9.2.3.1 Nếu dùng GRUB

Ví dụ bạn có hai phiên bản nhân trên máy 2.4.20 (phiên bản đang chạy) và 2.4.26 (phiên bản vừa được biên dịch) thì `grub.conf` có nội dung như sau:

```
default=0
timeout=20
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Linux (2.4.20)
    root (hd0,0)
    kernel /boot/vmlinuz-2.4.20 ro root=/dev/hda1
    initrd=/boot/initrd-2.4.20.img
```

Chỉnh thành:

```
default=0
timeout=20
splashimage=(hd0,0)/boot/grub/splash.xpm.gz

title Linux (2.4.26)
    root (hd0,0)
    kernel /boot/vmlinuz ro root=/dev/hda1
    initrd=/boot/initrd-2.4.26.img

title Linux (2.4.20)
    root (hd0,0)
```

```
    kernel /boot/vmlinuz-2.4.20 ro root=/dev/hda1
    initrd=/boot/initrd-2.4.20.img
```

Sau khi đã lưu trữ cấu hình của `/etc/grub.conf` ở trên (`/etc/grub.conf` là liên kết[30] đến `/boot/grub/menu.1st`) và khởi động lại máy để bắt đầu dùng nhân vừa được biên dịch. Nếu bạn dùng GRUB thì công tác biên dịch lại nhân Linux hoàn thành ở đây. Nên lưu ý trong ví dụ này, tôi dùng `vmlinuz` thay vì dùng `bzImage`, bạn nên chọn lựa theo ý và điều chỉnh cho phù hợp trong `grub.conf`.

Đối với giải pháp đề phòng trường hợp không thể dùng GRUB để khởi động vào nhân mới, xem chi tiết ở phần 9.1.1 ở trên.

#### 9.2.3.2 Nếu dùng LILO

Ví dụ bạn có hai phiên bản nhân trên máy 2.4.20 (phiên bản đang chạy) và 2.4.26 (phiên bản vừa được biên dịch) thì `lilo.conf` tương tự như sau:

```
prompt
timeout=50
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message

image=/boot/vmlinuz
initrd=/boot/initrd-2.4.20.img
root=/dev/hda1
label=linux
read-only
```

Chỉnh `/etc/lilo.conf` để cài nhân mới (2.4.26 cho ví dụ ở đây), bạn có `/etc/lilo.conf` như sau:

```
prompt
timeout=50
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
```

[30] symbolic link

```
image=/boot/vmlinuz-2.4.20
initrd=/boot/initrd-2.4.20.img
root=/dev/hda1
label=linux-2.4.20
read-only

image=/boot/vmlinuz
initrd=/boot/initrd-2.4.26.img
root=/dev/hda1
label=linux
read-only
```

Sau khi đã lưu trữ (save) cấu hình của `/etc/lilo.conf` ở trên, chạy lệnh:

```
# /sbin/lilo
```

để đăng ký nhân mới cho LILO.

Đối với giải pháp đề phòng trường hợp không thể dùng LILO để khởi động vào nhân mới, xem chi tiết ở phần 9.1.2 ở trên.

# 10 Khởi động lại máy và chỉnh lý nếu gặp trục trặc

Tới đây, bạn cần khởi động lại máy với nhân mới. Nếu không có gì trở ngại, máy sẽ khởi động vào Linux bình thường. Bạn có thể kiểm lại xem nhân mình đang chạy có đúng phiên bản vừa được biên dịch hay không bằng cách chạy:

```
$ uname -r
```

hoặc,

```
$ cat /proc/version
```

Nếu kết quả báo phiên bản nhân cũ thì có nghĩa trình khởi động (LILO hoặc GRUB) đã không khởi động nhân vừa được biên dịch. Bạn nên kiểm tra lại các file cấu hình (`lilo.conf` hoặc `grub.conf`) cho đúng theo chi tiết đã nêu ở trên.

## 10.1 Bị treo khi khởi động vào linux

Trở ngại trong giai đoạn khởi động vào Linux thông thường do cấu hình trình khởi động không đúng và trình khởi động được cài lên không đúng phân vùng khởi động[31] hoặc

[31] boot partition

MBR bị hỏng (hiếm thấy trong quá trình biên dịch lại và cài nhân mới nếu thực hiện đúng quy cách). Trong trường hợp này, bạn phải:

- dùng đến đĩa mềm "cấp cứu" được tạo ở phần ?? để khởi động vào Linux
- Đến giai đoạn này bạn hẳn phải biết vị trí của phân vùng gốc[32] (/) trên đĩa cứng để kết nối phân vùng[33] của đĩa cứng:

```
# mount /dev/hdXy /mount/point/somewhere
```

  trong đó X là ví trí đĩa cứng trên máy, y là vị trí phân vùng gốc trên đĩa cứng này.

- đổi root (`chroot`) trở thành phân vùng gốc của đĩa cứng:

```
# chroot /mount/point/somewhere
```

  trong đó `/mount/point/somewhere` là nơi đĩa cứng của bạn được kết nối[34].

- kiểm tra lại cấu hình của trình khởi động và cài đặt lại cho máy (xem phần 9.1.1 hoặc 9.1.2 tùy theo trình khởi động bạn dùng là GRUB hay LILO). Điểm cần chú ý ở đây cho GRUB là bạn phải chạy lệnh:

```
# /sbin/grub-install /dev/hdX
```

  trong đó `/dev/hdX` là tên thiết bị đĩa chứa MBR cho hệ thống (thường là đĩa đầu tiên trên máy[35]). Lệnh trên sẽ thiết lập lại bản ghi khởi động[36] và loại bỏ các trường hợp MBR bị hỏng. Tương tự cho LILO, bạn phải chạy lệnh:

```
# /sbin/lilo.
```

## 10.2 Bị treo trong quá trình nhân được load

Nếu bạn vướng vào các trở ngại trong giai đoạn nhân được tải lên thông thường là do các *drivers* tối cần thiết để *mount filesystems* trên máy bị thiếu. Giả sử bạn dùng ext3 cho phân vùng gốc[37] (/) chứa nhân. Để có thể kết nối phân vùng gốc[38] này, modul ext3 phải được biên dịch và `initrd` phải tải *module* này lên. Tương tự ứng dụng cho các trường hợp dùng *filesystem* khác và cũng thiếu module.

---

[32] root partition
[33] mount partition
[34] mount
[35] Primary Master
[36] boot record
[37] root partition
[38] mount root partition

Trong trường hợp này, bạn cần ghi phần lỗi được báo trong khi khởi động vào nhân mới để xác định lỗi này thuộc phần nào của cấu hình biên dịch nhân Linux và từ đó điều chỉnh lại và biên dịch lại cho thích ứng. Nói một cách tổng quát, bạn phải:

- khởi động lại máy vào phiên bản cũ của kernel (hoặc khởi động vào phiên bản cũ của nhân nếu bạn dùng biện pháp dự phòng đã được đề cập ở phần 9.1.1 và 9.1.2 ở trên)
- chọn lựa và chỉnh định cấu hình biên dịch nhân Linux lại (xem phần 7.2.2 để tránh lặp lại bước lựa chọn cấu hình một cách không cần thiết).
- thực hiện lại các bước đã nêu ra trong phần 8 và 9 ở trên

# 11 Vá và biên dịch nhân

Mã nguồn của nhân Linux thường được "vá" rồi biên dịch lại nhiều hơn là được biên dịch từ trọn bộ mã nguồn tải về từ http://sources.redhat.com/bzip2/kernel nếu bạn đã quen thuộc với quy trình tái biên dịch hoặc bạn có nhu cầu phải cập nhập kernel của máy thường xuyên. Tại sao lại cần "vá"? Mã nguồn của Linux kernel cần được vá vì các lý do thường gặp như sau:

- mã nguồn của nhân Linux kernel cập nhật. Bạn đã có sẵn mã nguồn của nhân Linux (cũ hơn) trên máy. Muốn nâng cấp phiên bản nhân của Linux, bạn chỉ cần tải các "miếng vá[39]" về để vá (thay vì phải tải trọn bộ mã nguồn của nhân Linux cho phiên bản mới).
- một số "*drivers*" được cập nhật. Để xử dụng các *driver* mới này (và các *drivers* này cần được biên dịch để nối với các thư viện hiện hành trên máy), bạn chỉ cần tải các "*miếng vá*" của những *drivers* này để vá nhân Linux và biên dịch lại chúng.

## 11.1 Các điểm quan trọng trước khi vá

Tương tự như phần 6.2, 6.3 và 6.4 ở trên, quy trình tải các miếng vá cho nhân Linux y hệt như tải trọn bộ gói mã nguồn của nhân Linux. Điểm khác biệt là bạn phải tải các hồ sơ khởi đầu bằng patch và chọn cho đúng các "*miếng vá*" cần thiết cho nhân cần được vá.

Điểm tối yếu cần ghi nhớ là khi vá mã nguồn của nhân Linux, bạn phải vá đúng thứ tự và đầy đủ các miếng vá cho đến đúng phiên bản cần có. Ví dụ, bạn đang có phiên bản nhân là 2.4.20 trên máy và bạn muốn biên dịch lại phiên bản kernel của máy trở thành 2.4.26. Thay vì tải trọn bộ mã nguồn của nhân 2.4.26 và biên dịch lại (như đã trình bày trong suốt bài viết này), bạn có thể tải các bản vá 2.4.21, 2.4.22, 2.4.23, 2.4.24, 2.4.25 và 2.4.26 về máy. Tổng cộng dung lượng các bản vá này chỉ là một phần rất nhỏ so với trọn bộ gói mã nguồn 2.4.26. Tất nhiên bạn đã có mã nguồn của kernel 2.4.20 trên máy.

[39] patches

## 11.2 Tải, xả và vá

Các miếng vá thường được nén ở hai dạng: `.gz` hoặc `.bz2` như gói mã nguồn. Bạn có thể tùy chọn và có thể tải các miếng vá này về bất cứ nơi nào trên máy. Sau khi tải chúng về, bạn có thể thực hiện quy trình tương tự như sau:

Giả định các bản vá được nén ở dạng `.bz2`, nơi chứa mã nguồn của nhân Linux ở `/usr/src` và thực tính của các miếng vá này đã được kiểm tra. Trong ví dụ này, giả định phiên bản đang dùng trên máy là 2.4.20 và phiên bản cần được vá sẽ là 2.4.26.

Chuyển vào thư mục `/usr/src`:

```
$ cd /usr/src
```

xả nén các miếng vá ở dạng `.bz2` vào thư mục `/usr/src`. Lặp lại cho đến khi xả hết các miếng vá:

```
$ tar xfvj /path/to/patch/patch-x.xx.xx ./
```

Dọn dẹp sạch sẽ mã nguồn nhân hiện có trên máy, giả định phiên bản mã nguồn hiện có là 2.4.20:

```
$ cd ./kernel-2.4.20
$ make mrproper
```

Lưu một bản mã nguồn kernel 2.4.20 trong thư mục `/usr/src` đề phòng cho sự cố trong quá trình vá (nếu bạn không lưu một bản mã nguồn nguyên thủy của nhân 2.4.20 trên máy, hoặc bản mã nguồn 2.4.20 này cũng đã được vá trước đây).

```
$ tar cvf ../linux-2.4.20.tar ./
```

Vá các miếng vá theo đúng thứ tự và theo dõi bất cứ lỗi nào được báo:

```
$ patch -p1 < ../patch-2.4.21
$ patch -p1 < ../patch-2.4.22
$ patch -p1 < ../patch-2.4.23
$ patch -p1 < ../patch-2.4.24
$ patch -p1 < ../patch-2.4.25
$ patch -p1 < ../patch-2.4.26
```

hoặc thực hiện kiểu "*lười*" như sau: tạo một biến môi trường `PATCH`[40] tạm thời chứa tên các miếng vá theo đúng thứ tự, tách rời bằng khoảng trống[41]:

[40] việc định nghĩa biến PATCH trong 2 dòng chỉ là thuận tiện khi dàn trang, tác giả dùng duy nhất 1 hàng. Kết quả hoàn toàn như nhau.

[41] space

```
$ export PATCH="patch-2.4.21 patch-2.4.22 patch-2.4.23 "
$ export PATCH="$PATCH patch-2.4.24 patch-2.4.25 patch-2.4.26"
```

Chạy vòng lặp:

```
$ for item in $PATCH; do patch -p1 < ../$item; done
```

Nếu trong khi vá không có gì trở ngại, bạn sẽ thấy các thông tin tương tự:

```
patching file xxx
patching file yyy
....
```

cho đến khi kết thúc.

Nếu trong khi vá bị báo lỗi, bạn phải ngưng bước vá (`Ctrl-C`) và kiểm tra xem bạn có dùng đúng bản vá và thực hiện các bản vá đúng thứ tự phiên bản hay không. Không nên tiếp tục với bước vá khi gặp lỗi vì chắc chắn bạn sẽ gặp trở ngại trong giai đoạn biên dịch sau này. Để tránh các trở ngại về sau, nếu bị báo lỗi trong khi vá, cách tốt nhất bạn nên xoá trọn bộ thư mục chứa mã nguồn của nhân Linux (đang được vá và bị lỗi) và xả gói mã nguồn nguyên thủy hoặc gói bạn vừa lưu trữ ở trên rồi thử lại.

Xoá thư mục chứa mã nguồn vừa vá và bị trục trặc, thư mục linux-2.4.20 được dùng như một ví dụ ở đây:

```
$ cd /usr/src
$ rm -rf ./linux-2.4.20
```

Xả gói mã nguồn được lưu trữ ở trên:

```
$ tar xvf linux-2.4.20
$ cd ./linux-2.4.20
```

và sau đó lặp lại bước vá theo đúng thứ tự các miếng vá.

Sau khi vá thành công, bạn nên thực hiện hai bước kế tiếp như sau trước khi bắt tay vào việc chuẩn bị cấu hình biên dịch mã nguồn nhân Linux:

Đổi tên thư mục chứa mã nguồn cho đúng phiên bản đã được vá (giúp bạn nhận diện phiên bản của mã nguồn đang có trên máy đã được vá tới phiên bản nào):

```
$ cd /usr/src
$ mv ./linux-2.4.20 ./linux-2.4.26
```

chỉnh giá trị "`VERSION`" trong file `Makefile` chính của mã nguồn Linux. Thư mục chứa mã nguồn lúc này đã được đổi tên thành `linux-2.4.26`: