The Linux BootPrompt-HowTo 中譯版

by Paul Gortmaker. v1.14, 1 February 1998

這份文件是啓動提示說明(BootPrompt-Howto)， 它整理了所有可以在啓動時期傳給 Linux 核心的啓動時期參數(arguments)。 包含所有核心與設備的參數。也包含關於核心如何排序(sorts) 啓動時期參數的討論， 還有一些啓動 Linux 核心常用軟體概述。

目錄

1 簡介

核心有能力在啓動時以‘指令列’方式接受有限的參數資訊， 類似執行程式時所給予的一些參數(argument list)。 通常這是用來提供核心一些它沒有辦法自己確定的硬體參數資訊， 或是避免變更(override)核心自己偵測所獲得的值。

然而，如果你直接把核心映像(kernel image)拷貝到軟碟上，（例如 cp zImage /dev/fd0）那麼你將沒有機會指定任何參數給核心。 所以大部份的 Linux 使用者都使用像是 LILO 或是 loadlin 一類能夠把這些參數傳給核心， 然後啓動它的軟體。

模組*(MODULE)*使用者請注意： 典型的啓動提示參數只能用於直接編譯到核心裡的硬體驅動程式。 它們對使用模組方式載入的驅動程式沒有影響。 大部份的發行套件(distributions) 都使用模組。 如果你無法確定， 可以對照 `/etc/conf.modules` 的內容查閱 `man depmod` 以及 `man modprobe`。

目前這版說明文件涵蓋到 v2.0.33 爲止的核心。同時也包含部份到 v2.1.84 爲止， 發展測試版核心的特性。

這份啓動提示說明的作者是：

Paul Gortmaker, `gpg109@rsphy1.anu.edu.au`

﹝請注意，特別給非 i386 機器或設備（尤其是 Atari/Amiga）用的 啓動提示參數目前並沒有寫進來。﹞

1.1 宣言與版權

這份文件並非絕對正確的眞理。然而它大概是你所能找到最新的資訊。 沒有人必須對你硬體所發生的事負責，除了你自己以外。 如果你的硬體冒煙燒掉了（・・・這幾乎不可能！）我也沒有責任。 也就是說根據這份文件中所提供的資訊去實行而造成的任何損害作者概不負責。

如果你想把這份文件集結出版，請與我聯絡，我會確認你取得的是否爲最新的資訊。 過去曾經有過時的 Linux 說明文件版本出版， 造成發展者爲了在新版文件中已有答案的問題想破頭殼。

This document is *not* gospel. However, it is probably the most up to date info that you will be able to find. Nobody is responsible for what happens to your hardware but yourself. If your hardware goes up in smoke (...nearly impossible!) I take no responsibility. ie. THE AUTHOR IS NOT RESPONSIBLE FOR ANY DAMAGES INCURRED DUE TO ACTIONS TAKEN BASED ON THE INFORMATION INCLUDED IN THIS DOCUMENT.

If you are intending to incorporate this document into a published work, please contact me, and I will make an effort to ensure that you have the most up to date information available. In the past, out of date versions of the Linux howto documents have been published, which caused the developers undue grief from being plagued with questions that were already answered in the up to date versions.

1.2 相關文件

最新版的文件永遠是核心的原始程式碼本身。等一下！先不要害怕。 閱讀原始檔裡的註解不需要懂任何程式設計。 例如，如果你要找可以傳給 AHA1542 SCSI 驅動程式的參數， 那就到 `linux/drivers/scsi` 目錄下去看 `aha1542.c` 這個檔， 在前面 100 行裡就可以找到一段原文說明此 1542 驅動程式所接受的啓動時期參數。

下個絕佳的參考是核心本身隨附的任何文件檔。目前並不多， 大部份可以在 `linux/Documentation` 目錄及其下的子目錄中找到。 `linux` 目錄通常可以在 `/usr/src/` 下找到。 有時候也可以在相關驅動程式目錄裡找到 `README.foo` 檔案 （例如 `linux/drivers/XXX/`，其中的 `XXX` 可能是 `scsi`, `char`, or `net`）。

如果你已經找到想要使用的啓動參數，而現在想知道如何把這些資訊傳給核心的話， 那就看看你用來啓動核心的軟體（例如 LILO 或者是 loadlin ）所隨附的文件。 下面有簡短的概述，但不能替代啓動軟體所隨附的文件。

1.3 Linux 新聞群組

如果你有關於傳遞參數給核心這方面的問題，請先讀一讀這份文件。 如果它以及前面所提到的相關文件都沒能回答你的問題，那麼你可以試試 Linux 新聞群組。 當然，在輕率的貼出你的問題前應該先閱讀該新聞群組， 因爲可能已經有人問過這個問題，或者這可能已經是個常見問題(a FAQ)。 張貼之前先快速瀏覽 linux 常見問題會是個好習慣。 你應該能夠在鄰近這份文件的某處找到常見問題集。

關於如何配置系統的一般性問題應該直接貼到 comp.os.linux.setup 新聞群組。 請你尊重這個有關張貼內容的一般慣例，並且， 不要把你的需求交叉張貼(cross-post)到其它群組去。

1.4 新版文件

新版文件可由匿名檔案傳輸服務(anonymous FTP)從 sunsite.unc.edu 這個站取得， 在 `/pub/Linux/docs/HOWTO/` 目錄裡。 注意，*SunSITE* 的負載經常很大， 所以建議你最好找個 Linux 檔案傳輸映射(mirror)節點取得此文件。 有新的資訊以及或是驅動程式時就會補充資料。 如果你現在讀的這份拷貝已經有幾個月之久，你可能得確認一下是否有新的。

這份文件係使用一套特別爲 Linux 說明文件計畫修改的 SGML 系統所產生的， 有各種不同的輸出格式，包括 postscript, dvi, ascii, html, 以及將要出現的 Texinfo。 我建議以 html（經由全球資訊網瀏覽器）或 Postscript/dvi 格式閱讀。 這兩者都包含 ascii 轉換過程中失去的交互參考(cross-references)。

如果你想從 sunsite 取得正式的拷貝，位址(URL)在此。 *BootPrompt-HOWTO* <http://sunsite.unc.edu/mdw/HOWTO/BootPrompt-HOWTO.html>

2 啓動提示參數概觀

這一節介紹一些可以用來傳遞啓動時期參數給核心本身的軟體範例。 也給你有關這些參數係如何被處理，啓動參數有什麼樣的限制， 以及它們如何被轉送到各個適當設備的觀念。

有一點很重要得注意的是在啓動參數中不應該使用空白， 只有在各別的參數之間可以。 單一參數其值的列表(A list of values)是在各值之間以逗號格開的，再一次， 沒有任何空白。參見下面的範例。

```
        ether=9,0x300,0xd0000,0xd4000,eth0  root=/dev/hda1
*RIGHT*
        ether = 9, 0x300, 0xd0000, 0xd4000, eth0  root = /dev/hda1
*WRONG*
```

2.1 LILO (LInux LOader)

最常用的是 Werner Almesberger 所寫的 LILO(LInux LOader) 程式。 它有能力啓動不同的核心，而且配置資訊存放在一個純文字檔裡。 大部份的發行套件都附有 LILO 作爲預設的啓動載入程式(boot-loader)。 LILO 可以一視同仁地啓動 DOS, OS/2, Linux, FreeBSD 等等，而且相當有彈性。

在你開機之後，典型的配置會讓 LILO 停住並印出 `LILO:`。 然後等個幾秒看看使用者有沒有輸入任何選項，沒有的話就啓動預設的系統。 在 LILO 配置檔裡典型的系統標籤是 `linux` 以及 `backup` 和 `msdos`。 如果你想要輸入啓動參數，可以打在這，在輸入你要 LILO 啓動的系統之標籤後面， 像下面這個範例所顯示的。

```
        LILO: linux root=/dev/hda1
```

LILO 隨附有絕佳的文件，而且爲了這裡所討論的啓動參數之使用， 當你想要把啓動時期參數加到 LILO 配置檔裡永久使用時 LILO 的 `append=` 指令就非常重要。 只要簡單地在 `/etc/lilo.conf` 檔中加入像 `append = "foo=bar"` 這樣的東東即可。 可以加在該檔的最前面，使它對每一節區(section)都生效， 或是加在某 `image=` 節區裡使它對該節區生效。 請參閱 LILO 文件裡更完整的說明。

2.2 LoadLin

另一個常用的 Linux 載入程式 'LoadLin' 則是一支有能力從 DOS 提示符號(prompt)下啓動 Linux 核心（可以給予啓動參數）的 DOS 程式， 假如某些系統資源可用的話就行。 這對使用 DOS 並想從 DOS 下啓動進入 Linux 的人而言很好用。

如果你擁有一些需要使用其所提供的 DOS 驅動程式來初使化以進入可用狀態 (known state)的硬體那麼它也很有用。 一個常見的例子是需要使用 DOS 驅動程式設定(twiddle) 幾個神秘的暫存器使它進入聲霸卡相容模式的”聲霸卡相容”音效卡。 啓動 DOS 與所附的驅動程式， 然後從 DOS 提示符號下

使用 LOADLIN.EXE 載入 Linux 以避免重新啓動(reboot)會重置(reset)該卡。 如此該卡維持在聲霸卡相容模式也就能在 Linux 下使用。

有一些其它程式可以用來啓動 Linux。 完整的列表請找找在你當地 Linux 檔案傳輸映設節點裡的程式， 在 system/Linux-boot/ 下。

2.3 “rdev” 公用程式

核心本身裡面的數個位元組存有少數幾個核心啓動參數的預設值。 在大部份的系統裡安裝有一支稱爲 rdev 的公用程式知道這些值在那裡， 以及如何改變它們。 它也能改變另外一些沒有相對之核心啓動參數的東西， 像是預設的顯示模式(video mode)。

此 rdev 公用程式通常也另稱(aliased to) swapdev, ramsize, vidmode 以及 rootflags。它能夠改變五種東西，分別是根目錄設備(root device)， 置換設備(swap device)，記憶體磁碟(RAM disk)之參數，預設的顯示模式， 及根目錄設備的存取設定(readonly/readwrite)。

更多有關 rdev 的資訊可以藉由輸入 rdev -h 或是參閱其線上說明頁 (man rdev)獲得。

2.4 核心如何排列參數

大部份的啓動參數格式爲：

```
name[=value_1][,value_2]...[,value_11]
```

其中 ‘name’ 是一個用來唯一識別(unique)相關的值（如果有的話） 是要給核心的那個部份的關鍵字。 多個啓動參數只是上述格式以空白分隔的列表。 注意那個 11 是眞實的限制， 因爲現行的程式碼對每個關鍵字只處理 11 個以逗號分隔的參數。 （然而，在非比尋常複雜的情況下你可以重覆使用相同的關鍵字加入 11 個以上的參數， 假如設定函式支援這樣搞的話。） 同時也要注意，核心把列表分成(splits)最多 10 個的整數參數， 後面並跟著一個字串，所以你不能眞的給它 11 個整數， 除非你自行在驅動程式中將第 11 個參數由字串轉換成整數。

大部份排序的動作是在 linux/init/main.c 裡面。 首先，核心查看參數是否爲 ‘root=’, ‘Ro’, ‘Rw’, or ‘Debug’ 這些特別的參數。 這些特別參數的意義本文中會進一步說明。

然後它走過(walks)一系列的設定函式（放在 bootsetups 陣列裡） 看看所指定的參數字串（像是 ‘foo’）是不是與其中的一個設定函式（foo_setup()）相關， 以設定特定設備或核心的一部份。 如果你傳給核心 foo=3,4,5,6,bar 這樣一行那麼核心會搜尋 bootsetups 陣列看看 ‘foo’ 是否有註冊。 如果有，那麼它會呼叫與 ‘foo’ 相關聯的設定函式（foo_setup()） 並交給它在核心指令列所給的整數參數 3, 4, 5, and 6，並交給它字串參數 bar。

2.5 設定環境變數

任何不被接受當作上述設定函式的 ‘foo=bar’ 型式之敘述會被解譯爲要設定的環境變數。 一個（沒用的？）範例是以 ‘TERM=vt100’ 作爲啓動參數。

2.6 傳遞參數給 ‘init’ 程式

任何剩下的， 核心不管而且不被解譯成環境變數之參數接下來會傳給第一個程序(process)， 通常是 init 程式。 最常傳給 init 程序的參數是 *single* 這個字， 指示 init 把電腦啓動在單人模式(single user mode) 並且不啓動所有一般的駐留程式(daemons)。 查閱你系統裡安裝的 `init` 版本之線上手冊看看它接受那些參數。

3 一般無關設備特性的啓動參數

有些啓動參數並非與任何設備或周邊相關。 它們乃是與某個核心內部參數相關，像是記憶體的處理，記憶體磁碟的處理， 根檔案系統的處理以及其它東東。

3.1 根檔案系統選項

下列選項皆與核心如何選擇及處理根檔案系統有關。

3.1.1 **'root='** 參數

此參數告訴核心啓動時以那個設備作爲根檔案系統使用。 此設定的預設值爲建造(bulid)核心時系統的根檔案系統設備。 例如，如果核心是在一個以 '/dev/hda1' 作爲根檔案系統分割區的系統上建造的，那麼預設的根檔案系統設備就是 'dev/hda1'。 要變更此預設值，並選擇第二台軟碟機作爲根檔案系統設備的話， 可以使用 'root=/dev/fd1'。

可用的根檔案系統設備爲下列設備其中之一：

(1) /dev/hdaN to /dev/hddN, 這是 ST-506 相容磁碟 'a to d' 上的第 N 個分割區。

(2) /dev/sdaN to /dev/sdeN, 這是 SCSI 相容磁碟 'a to e' 上的第 N 個分割區。

(3) /dev/sdaN to /dev/xdbN, 這是 XT 相容磁碟 'a to b' 上的第 N 個分割區。

(4) /dev/fdN, 這是軟碟機號碼 N。 N=0 爲 DOS 下的 'A:' 磁碟機，而 N=1 則爲 'B:'。

(5) /dev/nfs, 這並非真的是個設備， 而是一個告訴核心經由網路取得根檔案系統的旗標。

上述磁碟設備的另一種數字格式， 更爲笨拙且更沒有可攜性的主要次要(major/minor) 號碼也能接受。 （例如 /dev/sda3 的主要號碼爲 8，次要號碼爲 3， 所以你可以使用 `root=0x803` 作爲另一種指定方式。）

這是少數幾個在核心中存有預設值， 所以可以用 `rdev` 公用程式改變的核心啓動參數。

3.1.2 **'ro'** 參數

當核心啓動時它需要有個根檔案系統以讀取一些基本的東西。 這就是掛在根目錄的根檔案系統。 然而，如果根檔案系統以可以寫入的方式掛上來的話， 你就無法在有檔案寫到一半的情況下確實地檢查檔案系統的完整性。 此 'ro' 選項告訴核心以唯讀的方式掛入根檔案系統， 如此任何檔案系統一致性檢查程式(fsck) 在執行檢查時都可以安全地假設沒有檔案寫到一半的情況。 沒有任何程式或程序可以寫入此檔案系統直到它重新以可讀寫的方式掛入爲止。

這是少數幾個在核心中存有預設值， 所以可以用 `rdev` 公用程式改變的核心啓動參數。

3.1.3 'rw' 參數

這與上一個完全相反，它告訴核心以可讀取寫入的方式掛入根檔案系統。 無論如何，預設是以可讀寫方式掛入根檔案系統。 不要在以可讀寫方式掛入的檔案系統上執行任何的 'fsck' 程式。

參數使用的值與上一個儲存於核心映像的參數相同，可經由 `rdev` 存取。

3.2 與記憶體磁碟(RAM Disk)管理有關的選項

下列選項皆與核心如何處理記憶體磁碟設備有關， 這種虛擬磁碟設備通常在安裝階段啓動機器時使用， 或配合使用模組化驅動程式以存取根檔案系統的機器上。

3.2.1 'ramdisk_start=' 參數

爲了使核心映像能夠與壓縮的記憶體磁碟映像放在一張軟碟內， 所以加入這個 'ramdisk_start=<offset>' 指令。 核心不能夠放在壓縮過的記憶體磁碟之檔案系統映像裡， 因爲它得從最開始的第零磁區開始放置， 如此基本輸出入系統(BIOS)才能載入啓動磁區而核心也才能夠開始啓動執行。

注意：如果你使用的是沒有壓縮的記憶體磁碟映像， 那麼核心可以是要載入記憶體磁碟之檔案系統映像的一部份， 且該軟碟可以由 LILO 啓動，兩者也可以如同壓縮之映像般爲分開的兩部份。

如果你使用啓動根(boot/root)兩張磁片的方式（核心一張， 記憶體磁碟映像放第二張）那麼記憶體磁碟會由第零磁區開始， 並使用零作爲偏移值(offset)。因爲這是預設值，你根本不必眞的去使用這個指令。

3.2.2 'load_ramdisk=' 參數

此參數告訴核心是否要載入一份記憶體磁碟映像。 'load_ramdisk=1' 指定核心將軟碟載入至記憶體磁碟中。 預設值爲零，表示核心不應該試著去載入記憶體磁碟。

請參閱 `linux/Documentation/ramdisk.txt` 檔案中新的啓動參數與如何使用它們的完整敘述。 它同時也說明如何藉由 'rdev' 來設定及儲存這些個參數。

3.2.3 'prompt_ramdisk=' 參數

此參數告訴核心是否要給你個提示要求插入含記憶體磁碟映像的磁片。

在只用一張軟碟的配置下記憶體磁碟映像與剛剛載入啓動的核心在相同的軟碟上故不需要提示。 這種情況可以用 'prompt_ramdisk=0'。 在使用兩張軟碟的配置下你需要有個抽換磁片的機會，故可以使用 'prompt_ramdisk=1'。 因爲這是預設值，所以不必眞的去指定它。 （軼聞：從前人們習慣使用 'vga=ask' 這個 LILO 選項來暫時停止啓動程序以取得抽換啓動磁片及根磁片的機會。）

請參閱 `linux/Documentation/ramdisk.txt` 檔案中新的啓動參數與如何使用它們的完整敘述。 它同時也說明如何藉由 'rdev' 來設定及儲存這些個參數。

3.2.4 'ramdisk_size=' 參數

因爲記憶體磁碟實際上會依需求動態成長， 所以其大小有個上限加以限制以免它用光所有可用的記憶體而壞事。 預設值 4096(i.e. 4MB) 應該足夠滿足大部份的需求。 你可以用這個啓動參數變更此預設值

。

參閱 linux/Documentation/ramdisk.txt 檔案中新的啟動參數與如何使用它們的完整敘述。 它同時也說明如何藉由 'rdev' 來設定及儲存這些個參數。

3.2.5 'ramdisk=' 參數（過氣了）

（注意：這個參數是舊的，除了 v1.3.47 版以及更舊的的核心之外不應該使用。 應該使用的是前面所述的指令）

這個參數以千位元組(kB)爲單位指定記憶體磁碟設備的大小。 例如，如果想要把位於一張 1.44MB 軟碟上的根檔案系統載入至記憶體磁碟設備， 可以用：

```
ramdisk=1440
```

這是少數幾個在核心中存有預設值，所以可以用 rdev 公用程式改變的核心啟動參數。

3.2.6 'noinitrd'（啓始記憶體磁碟）參數

v2.x 及更新的核心有個特性，根檔案系統最初是記憶體磁碟， 然後核心執行記憶體映像上的 /linuxrc 。 這個特性典型係用於載入某些掛入眞正的根檔案系統所需要的模組 （例如，載入儲存於記憶體磁碟映像裡的 SCSI 驅動程式， 然後掛入在某 SCSI 磁碟裡眞正的根檔案系統。）

實際的 'noinitrd' 參數決定核心啓動之後如何處理 initrd 資料。 如果有指定，它可以經由 /dev/initrd 存取， 可以在記憶體釋放回系統之前讀取一次。 有關初始之記憶體磁碟的完整細節與使用請參閱 linux/Documentation/initrd.txt。 此外，最新版的 LILO 及 LOADLIN 應該會包含其它有用的資訊。

3.3 與記憶體管理有關的參數

下列參數會改變 linux 偵測或處理系統實體及虛擬記憶體的方式。

3.3.1 'mem=' 參數

這個參數有兩個目的：原先的目的是指定機器所安裝的記憶體數量 （如果你想限制 linux 能使用的記憶體數量可以指定一個較小的值）。 第二個（很少用）目的則是指定 mem=nopentium 以便告訴 linux 核心不要使用 4MB 分頁表(page table)這個效能特性。

在個人電腦規格中， 原先定義回傳安裝記憶體數量的基本輸出入呼叫被設計成最多只能回報最多 64MB 。 （是的，另一個缺乏遠見的設計，就像 1024 磁簇的磁碟限制・・・哎。） Linux 在啓動時會使用此基本輸出入呼叫以確定安裝的記憶體數量。 如果你安裝的記憶體超過 64MB，可以用這個啓動參數告訴 Linux 你有多少記憶體。 下面引用 Linus 對 mem= 參數的說明。

” 核心會接受任何你所給予的 'mem=xx' 參數，而如果它發現你騙它， 那它遲早一定會當的很難看。 這個參數指示可以定址的最高記憶體位址，所以， 例如 'mem=0x1000000' 表示你有 16MB 的記憶體。 對擁有 96MB 的機器而言此值爲 'mem=0x6000000'。

注意注意注意：

某些機器可能會使用最上層(top)的記憶體作爲基本輸出入系統的快取等等一類用途， 所以你可能並非實際擁有 96MB 的可定址空間。反之亦然：

某些晶片組會將基本輸出入系統涵蓋的實體記憶體區域對應(map)到最上層記憶體後面， 所以記憶體最上層實際上可能是 96MB + 384kB。 如果你告訴 linux 的記憶體比它實際上擁有的還多，那就會出狀況：也許不會立刻出事， 但最後一定會發生。”

注意，此參數並非一定得是十六進位型態， 而且可以使用 ‘k’ 與 ‘M’（大小寫無關）字尾分別指定千位元組以及百萬位元組。 （‘k’ 會把你給的值旋轉(shift) 10 位元，而 ‘M’ 會旋轉 20 位元。） 上述警告依然沒變，沿上例， 一台 96MB 的機器也許可以設 `mem=97920k` 而無法使用 `mem=98304k` 或 `mem=96M`。

3.3.2 ‘swap=’ 參數

這允許使用者調整某些與磁碟置換(swapping)有關的虛擬記憶體參數(VM)。 它接受下列八個參數：

```
MAX_PAGE_AGE
PAGE_ADVANCE
PAGE_DECLINE
PAGE_INITIAL_AGE
AGE_CLUSTER_FRACT
AGE_CLUSTER_MIN
PAGEOUT_WEIGHT
BUFFEROUT_WEIGHT
```

有興趣的玩家建議你閱讀 `linux/mm/swap.c` 並注意 `/proc/sys/vm` 裡面的東西。

3.3.3 ‘buff=’ 參數

類似 ‘swap=’ 參數，這允許使用者調整某些與緩衝記憶體(buffer)之管理相關的參數。 它接受下列六個參數。

```
MAX_BUFF_AGE
BUFF_ADVANCE
BUFF_DECLINE
BUFF_INITIAL_AGE
BUFFEROUT_WEIGHT
BUFFERMEM_GRACE
```

有興趣的玩家建議你閱讀 `linux/mm/swap.c` 並注意 `/proc/sys/vm` 裡面的東西。

3.4 以網路檔案系統(NFS)作爲根檔案系統使用的參數

藉由透過網路檔案系統(NFS: Network FileSystem) 提供根檔案系統 Linux 支援無碟(diskless)工作站這樣的系統。 這些參數係用來告訴無碟工作站要從那一台機器取得其系統。 同時要注意 `root=/dev/nfs` 參數是必需的。 有關使用網路檔案系統作爲根檔案系統之細節說明在 `linux/Documentation/nfsroot.txt` 檔案裡。 你應該閱讀這個檔案， 因爲下面只是從那個檔案直接拿過來的快速摘要。

3.4.1 **'nfsroot='** 參數

這個參數告訴核心以那一台機器， 那個目錄以及那些個網路檔案系統選項作爲根檔案系統使用。 此參數的格式如下：

```
nfsroot=[<server-ip>:]<root-dir>[,<nfs-options>]
```

如果指令列上沒有給定 nfsroot 參數，則將使用 '/tftpboot/%s' 預設值。 其它選項如下：

<server-ip> – 指定網路檔案系統伺服端的網際網路位址(IP address)。 如果沒有給定此欄位，則使用由 nfsaddrs 變數（見下面）所決定的值。 此參數的用途之一是允許使用不同機器作爲反向位址解析協定 (RARP) 及網路檔案系統伺服端。通常你可以不管它（設爲空白）。

<root-dir> – 伺服端上要作爲根掛入的目錄名稱。如果字串中有個 '%s' 符記(token)， 此符記將代換爲客户端網際網路位址之 ASCII 表示法。

<nfs-options> – 標準的網路檔案系統選項。所有選項都以逗號分開。 如果沒有給定此選項欄位則使用下列的預設值：

```
port            = as given by server portmap daemon
rsize           = 1024
wsize           = 1024
timeo           = 7
retrans         = 3
acregmin        = 3
acregmax        = 60
acdirmin        = 30
acdirmax        = 60
flags           = hard, nointr, noposix, cto, ac
```

3.4.2 **'nfsaddrs='** 參數

這個啓動參數設定網路通訊所需的各種網路界面位址。如果沒有給定這個參數， 則核心會試著使用反向位址解析協定以及或是啓動協定(BOOTP)以找出這些參數。 其格式如下：

```
nfsaddrs=<my-ip>:<serv-ip>:<gw-ip>:<netmask>:<name>:<dev>:<auto>
```

<my-ip> – 客户端的網際網路位址。如果沒設， 此位址將由反向位址解析協定或啓動協定來決定。 使用何種協定端視配置核心時打開的選項以及 <auto> 參數而定。 如果設定此參數，就不會使用反向位址解析協定或啓動協定。

<serv-ip> – 網路檔案系統伺服端之網際網路位址。 如果使用反向位址解析協定來決定客户端位址並且設定此參數， 則只接受從指定之伺服端傳來的回應。 要使用不同的機器作爲反向位址解析與網路檔案系統伺服端的話， 在此指定你的反向位址解析協定伺服端（保持空白）並在 nfsroot 參數（見上述） 中指定你的網路檔案系統伺服端。 如果此項目空白則使用回答反向位址解析協定或啓動協定之伺服端的位址。

<gw-ip> – 閘道(gateway)之網際網路位址，若伺服端位於不同的子網路上時。 如果此項目空白則不使用任何閘道並假設伺服端在本地的(local)網路上， 除非由啓動協定接收到值。

<netmask> – 本地網路介面的網路遮罩。 如果爲空白，則網路遮罩由客户端的網際網路位址導出，除非由啓動協定接收到值。

<name> – 客户端的名稱。 如果空白，則使用客户端網際網路位址之 ASCII-標記法，或由啓動協定接收的值。

<dev> – 要使用的網路設備名稱。 如果爲空白，所有設備都會用來發出反向位址解析請求， 啓動協定請求由最先找到的設備發出。 網路檔案系統使用接收到反向位址解析協定或啓動協定回應的設備。如果你只有一個設備那你可以不管它。

<auto> – 用以作爲自動配置的方法。 如果是 ‘rarp’ 或是 ‘bootp’ 則使用所指示的協定。 如果此值爲 ‘both’ 或空白，若配置核心時有打開這兩種協定則都使用。 ‘none’ 表示不使用自動配置。這種情況下你必須指定前述欄位中所有必要的值。

此 <auto> 參數可以作爲 nfsaddrs 的參數單獨使用（前面沒有任何 ‘:‘ 字元）， 這種情況下會使用自動配置。然而，此種情況不能使用 ‘none’作爲值。

3.5 其它另外的核心啓動參數

這些啓動參數讓使用者調整某些核心內部的參數。

3.5.1 The ‘debug’ Argument

核心經由 `printk()` 函式傳達重要（以及沒那麼重要）的訊息給管理者(operator)。 如果訊息很重要，`printk()` 函式也會顯示到主控台(console)上， 就如同丟給 `klogd()` 記錄到磁碟上一般。 如同記錄到磁碟上一般在主控台上顯示重要訊息的原因是因爲在不幸的狀況下 （例如磁碟損壞）訊息將無法存入磁碟而漏失。

到底什麼重要什麼不重要其門檻藉由 `console_loglevel` 變數設定。 預設是把任何比 `DEBUG`（等級 7）重要的東西記錄到主控台上去。 （這些等級定義在 `kernel.h` 含入檔中）指定 `debug` 啓動參數將使主控台的記錄等級設爲 10， 所以所有核心訊息都會出現在主控台上。

主控台記錄等級通常也可以在執行期間經由 `klogd()` 程式選項設定。 查看你系統之安裝版本的線上說明頁看看該怎麼做。

3.5.2 ‘init=’ 參數

核心啓動時預設執行 ‘init’ 程式，這支程式接下來經由執行 getty 程式， 跑 ‘rc’ 指令稿(scripts)以及諸如此類的東東爲使用者設定電腦。 核心首先尋找 `/sbin/init`，然後是 `/etc/init` (depreciated)， 而最後它會去試 `/bin/sh` （可能在 `/etc/rc` ）。 如果說，例如，如果你的 init 程式壞掉了，只要使用 `init=/bin/sh`

這個啓動參數就能讓你在啓動時直接跳到解譯環境(shell)，使你能夠換掉壞掉的程式。

3.5.3 ‘no387’ 參數

某些 i387 協同處理器晶片有臭蟲(bugs)，在 32 位元保護模式下會出現。 例如，部份早期的 ULSI-387 晶片在執行浮點運算時會死當， 這似乎是因爲 FRSAV/FRRESTOR 指令的一隻蟲。 使用 ‘no387’ 啓動參數使 Linux 就算你眞的有數值輔助處理器也忽略它的存在。 當然編譯核心時你必須加入數值模

擬支援！ 如果你有某些能夠使用 80287 浮點處理單元(FPU)的古董級 386 機器那這也很有用， 因爲 linux 無法使用 80287。

3.5.4 'no-hlt' 參數

i386（及其後繼者） 家族的中央處理器有個 'hlt' 指令用來告訴中央處理器現在閒閒沒事做， 直到某個外部設備（鍵盤，數據機，磁碟等等）呼叫中央處理器執行任務爲止。 這個指令會使中央處理器進入‘省電’模式， 像個死人(zombie)般坐在那裡直到有個外部設備叫它起來 （經由中斷(interrupt)，通常是）。 部份早期的 i486DX-100 晶片處理這個 'hlt' 指令有問題， 使用過這個指令後它們不能可靠地回到運作模式。 使用 'no-hlt' 這個指令告訴 Linux當無事可做時跑個無窮迴圈而不停住你的中央處理器。 讓有這些晶片的人們能夠使用 Linux，然而還是建議有可能的話就換掉這些晶片。

3.5.5 'no-scroll' 參數

啓動時使用這個參數可以關閉捲頁(scrolling)特性，這個特性使得 Braille 終端機難以使用。

3.5.6 'panic=' 參數

在不太可能發生的核心異常(kernel panic)事件下（像是核心偵測到內部錯誤， 並認爲這樣的錯誤嚴重到應該發出警訊並停止任何作業），

預設的處理方式是就停在那兒直到有人過來並注意到螢幕上的異常警訊然後重新啓動機器。

然而如果機器是在無法觸及，隔離的地方跑的話也許會希望它能自動重置自己然後回到線上。 例如，啓動時使用 `panic=30` 會使核心在發生核心異常 30 秒後試著重新啓動自己。 若此值爲零則使用預設的處理方式，就是一直在那兒等。

注意，這個逾時(timeout)設定也可經由 `/proc/sys/kernel/panic` 系統控制 (sysctl)介面讀取與設定。

3.5.7 'profile=' 參數

核心發展者們可以打開某個選項以得知核心如何使用中央處理器時間以及用在那裏， 以便最大化效率(efficiency)與效能(performance)。 這個選項讓你在啓動時設定觀察變化的計數(the profile shift count)。此值一般設爲二。你也可以在編譯核心時打開此功能。無論那種情況，你都需要像 `readprofile.c` 這類可以處理 `/proc/profile` 輸出的工具。

3.5.8 'reboot=' 選項

這個選項控制重置電腦時 （典型是經由 `/sbin/init` 處理的 Control-Alt-Delete 動作） Linux 所做的重新啓動之種類。 新的 v2.0 核心預設的動作是做‘冷’開機 （完全重置，基本輸出入系統執行記憶體檢查等等）以代替‘暖’開機 （沒有完全重置，沒有記憶體檢查）。

改爲預設冷開機是爲了要在便宜的爛爛的，暖開機請求沒辦法重新啓動的硬體上工作。 可以使用 `reboot=w` 設爲原先的方式（暖開機），其實可以用任何以 `w` 開頭的字來設定。

爲什麼這會造成困擾？ 某些具有記憶體快取的磁碟控制器能夠感測到暖開機， 並且把所有暫存的資料寫到磁碟上。 冷開機可能會重設該卡，在快取卡記憶體裡面的回寫(write-back)資料就會漏失。 已經有人回報系統記憶體檢查很花時間以及或是小型電腦智慧介面基本輸出入系統 (SCSI BIOSes)冷開機時花較久的時間初始化是使用暖開機的好理由。

3.5.9 'reserve=' 參數

這是用來保護輸出入埠區域不要偵測。這個指令的格式是：

```
reserve=iobase,extent[,iobase,extent]...
```

在某些機器上也許必須避免設備驅動程式去檢查（自動偵測）在某些特殊區域的設備。 這可能是因爲硬體設計的不良而會使得啓動終止（像是某些乙太網路卡）， 會被誤認的硬體，狀態會因爲較前面的偵測而改變的硬體， 或者只是你不想讓核心初始化的硬體。

此 reserve 啓動時期參數藉由指定一段不要偵測的輸出入埠範圍以解決此問題。 此段區域在核心的輸出入埠註冊表格當中被視爲已經在該處找到設備 （名稱爲 reserved ）而保留。注意，絕大部份的機器都不需要此機制。 只有眞的有問題或特殊情況才會需要用到這東東。

位於指定之區域中的輸出入埠乃是靠著在偵測一段輸出入區域之前先執行 check_region() 以避免設備偵測。 這用於某些遇上 NE2000 會掛掉或者會誤認其它設備的驅動程式。

正確的設備驅動程式不應該去偵測保留區域，除非另一個啓動參數明確地指示它這樣做。 這意謂著 reserve 經常與其它啓動參數一起使用。 如果你指定一段保留區域以保護某特定設備的話， 你必須明確地指定此設備的偵測範圍， 大部份的驅動程式如果有給它們明確的位址就會忽略輸出入埠註冊表。

例如，此啓動列

```
reserve=0x300,32  blah=0x300
```

保留 0x300-0x31f 不讓除了 'blah' 的設備驅動程式外的所有驅動程式去偵側。

如同一般啓動時期所用的格式，它也有 11 個參數之限制， 因此每個 reserve 關鍵字只能指定 5 個保留區域。 如果你有異常複雜的需求，可以使用多重 reserve 指定。

3.5.10 'vga=' 參數

注意，這並不眞的是個啓動參數。它是由 LILO 解譯的選項， 與所有其它由核心處理的啓動參數並不相同。 然而因爲它的使用變得如此普遍故值得在此加上一筆。 這個選項也可以經由使用 rdev -v 或相同的 vidmode 指令對 vmlinuz 檔案作設定。 這使得設定程式碼能夠在眞的啓動 Linux 核心之前使用視訊(video)基本輸出入系統改變預設的顯示模式。 典型的模式是 80x50, 132x44 等等。 使用此選項最好的方式是以 vga=ask 啓動， 如此在啓動核心前會有個列表提示你的顯示卡可以使用的各種模式。 一旦你從上述列表得知你想使用的號碼，以後就可以把它放在 'ask' 的位置。 更進一步的資訊請參閱隨附於所有新版核心的 linux/Documentation/svga.txt 檔案。

注意，新的核心（2.1 版以上）有改變顯示模式的設定程式碼選項 *Video mode selection support*， 所以如果你想使用這個特性那麼你就得打開這個選項。

4 小型電腦智慧介面(SCSI)周邊的啓動參數

這一節包含用來傳遞有關小型電腦智慧介面(SCSI)卡(host adapters)及 SCSI 設備之資訊的啓動參數說明。

4.1 中層驅動程式的參數

中層驅動程式處理像磁碟，光碟及磁帶等等而不深入介面卡的特性。

4.1.1 最大偵測邏輯號碼(LUN)數(‘max_scsi_luns=’)

每個 SCSI 設備本身裡面都可以包含一個‘子設備’號碼。 最常見的例子是那些一次可以放好幾張片子的新型 SCSI 光碟機。 每張光碟由指定之設備的‘邏輯號碼(Logical Unit Number: LUN)’定址。 但大部份的設備，像硬碟機，磁帶機等等都是單一個的設備， 而邏輯號碼設爲零。

只有一個邏輯編號的設備若韌體寫得不好就會出問題。 某些設計不良的 SCSI 設備（舊的以及就那麼巧被你碰上的新設備）無法接受不爲零的邏輯編號偵測。 它們會掛在那，而且有可能把整個 SCSI 匯流排(bus)都帶著一起陪葬。

較新的核心有一個配置選項可以讓你設定所要偵測之最大邏輯編號。 預設只偵測到邏輯編號零以必免上述問題。

在啓動時指定要偵測的邏輯號碼可以鍵入 ‘max_scsi_luns=n’ 作爲啓動參數， 其中 n 是介於壹到捌的數字。 要避免上述問題可以使用 n=1 以避免這些個爛設備不高興。

4.1.2 SCSI 磁帶驅動程式的參數(‘st=’)

SCSI 磁帶驅動程式的某些啓動時期配置可以藉由使用下列參數達成：

```
st=buf_size[,write_threshold[,max_bufs]]
```

前面兩個數字的單位是千位元組。預設的 buf_size 是 32kB。 最多可以指定爲 16384kB，大的荒謬。 write_threshold 緩衝區確認送至磁帶的值，預設爲 30kB。 最大緩衝區個數視偵測到的裝置數量而定，預設值爲二。使用範例如：

```
st=32,30,2
```

完整的細節可以在 README.st 檔案中找到， 位於核心原始程式碼結構(tree)的 scsi 目錄裡。

4.2 SCSI 卡的參數

本節的常見標記：

iobase – SCSI 卡進駐的第一個輸出入埠。 以十六進制表示法指定，而且通常落在 0x200 到 0x3ff 之間。

irq – 介面卡設定使用的硬體中斷。 其可用的值與介面卡有關， 但通常是 5, 7, 9, 10, 11, 12, 以及 15。 其它值通常由一般的周邊設備使用， 像是 IDE 硬碟，軟碟，串列埠等等。

dma – 介面卡使用的直接記憶體存取(Direct Memory Access:DMA)通道。 典型只用於具匯流排主控(bus-mastering)能力的卡。 PCI 及 VLB 介面卡本即具匯流排主控，且不需要任何 ISA 直接記憶體存取通道。

scsi-id – 在 SCSI 匯流排上，介面卡用來識別它本身的識別號碼。 只有某些介面卡允許你改變這個值，大部份則都內定且永久不變。 通常預設值是七，但西捷(Seagate)以及 Future Domain TMC-950 卡使用六。

`parity` – SCSI 介面卡是否期望所連接的設備交換資訊時提供同位檢查碼。 指定一表示打開同位檢查，零則爲關閉。 同樣地，並非所有的介面卡都支援選用同位檢查這個啓動參數。

4.2.1 Adaptec aha151x, aha152x, aic6260, aic6360, SB16-SCSI ('aha152x=')

aha 編號對照卡而 aic 編號對照這些卡上實際的 SCSI 晶片， 包含 Soundblaster-16 SCSI。

偵測這些 SCSI 卡的程式碼尋找從卡上安裝進來的基本輸出入系統， 如果沒有，就找不到你的卡。然後你就得使用這種型式的啓動參數：

```
aha152x=iobase[,irq[,scsi-id[,reconnect[,parity]]]]
```

注意，如果此驅動程式編譯時有打開錯誤偵測，則可以指定第六個值以設定偵錯層級。

所有參數都如本節最前面所述，而 `reconnect` 值如果不爲零則允許設備離線重新連線。 使用範例如下：

```
        aha152x=0x340,11,7,1
```

注意，參數必須依序指定，意謂如果你想指定同位檢查， 那麼你也得指定 iobase, irq, scsi-id 以及 reconnect 值。

4.2.2 Adaptec aha154x ('aha1542=')

這些是 aha154x 系列的卡。 aha1542 系列的卡板子上有一個 i82077 軟碟控制器，而 aha1540 系列的卡沒有。

這些卡是匯流排主控卡，而且有可以設定”公平性”的參數，用來與其它設備共用匯流排。 其啓動參數看起來像下面這樣。

```
        aha1542=iobase[,buson,busoff[,dmaspeed]]
```

可用的 `iobase` 值通常是： `0x130`, `0x134`, `0x230`, `0x234`, `0x330`, `0x334` 其中之一。 仿(Clone)卡可能允許使用其它值。

`buson`, `busoff` 的值是該卡佔用工業標準架構(ISA)匯流排的毫秒數(microsecond)。 預設值是 11us on, and 4us off，如此其它的卡 （像是工業標準架構介面的 LANCE 乙太網路卡）就有機會存取工業標準架構匯流排。

`dmaspeed` 的值是進行直接記憶體存取的速率（以百萬位元組每秒爲單位）。 預設爲 5MB/s。 新版的卡可以讓你從軟體配置(soft-configuration)中選擇這個值， 較舊的卡則使用跳接(jumpers)。假設你的主機板夠力的話可以使用 10MB/s。 如果使用 5MB/s 以上的值請小心試驗。

4.2.3 Adaptec aha274x, aha284x, aic7xxx ('aic7xxx=')

這些卡可以接受此種格式的參數。

```
        aic7xxx=extended,no_reset
```

如果 `extended` 的值不爲零則表示打開對大容量磁碟的延伸轉換(extended translation)。 `no_reset` 值如果不爲零則是告訴驅動程式啓動時， 設定介面卡的時候不要重置 SCSI 匯流排。

4.2.4 AdvanSys SCSI Host Adaptors ('advansys=')

AdvanSys 驅動程式最多能接受四個輸出入位址以偵測 AdvanSys SCSI 卡。注意，這些值（若使用的話）完全不會影響 EISA 或是 PCI 的 偵測。它們僅用來偵測 ISA 以及 VLB 介面卡。還有，如果此驅動程式編譯時有打開錯誤偵測的話，那麼錯誤偵測的輸出輸出層級可以 藉由加入 0xdeb[0-f] 參數來設定。

4.2.5 Always IN2000 介面卡 ('in2000=')

跟其它 SCSI 卡的啓動參數不同， IN2000 驅動程式使用由美國標準交換碼(ASCII) 所組成的前置(prefix)字串代表大部份的整數參數。 這是一份支援列表：

ioport:addr – 其中 addr 是卡（通常沒有唯讀記憶體）的輸出入位址。

noreset – 沒有選用的參數。這避免啓動時期的 SCSI 匯流排重置動作。

nosync:x – x 是個位元遮罩(bitmask)，其中前面七個位元對應到七個 SCSI 設備 （第零個位元是第零號設備，餘類推）。

設定其中的位元以避免對該設備進行同步(sync)協商(negotiation)。 驅動程式預設是關閉所有設備之同步。

period:ns – ns 是以奈秒(nanoseconds)爲單位之最小 SCSI 資料傳輸時間區段 (period)。預設爲 500; 可接受的值是 250 到 1000。

disconnect:x – x = 0 即絕不允許離線，2 即一定可以離線。 x = 1 則爲’適當’時可離線，這是預設值， 而且是一般情況的最佳選擇。

debug:x 如果定義了 'DEBUGGING_ON' 則 x 就是設定不同錯誤偵測輸出的位元遮罩 —參見 in2000.h 中定義的 DB_xxx。

proc:x – 如果定義了 'PROC_INTERFACE' 則 x 就是決定 /proc 介面運作方式及功能的位元遮罩—參見 in2000.h 中定義的 PR_xxx。

下面列出一些使用範例：

```
in2000=ioport:0x220,noreset
in2000=period:250,disconnect:2,nosync:0x03
in2000=debug:0x1e
in2000=proc:3
```

4.2.6 使用 AMD AM53C974 的硬體 ('AM53C974=')

與其它驅動程式不同，這一個不使用啓動參數來傳達輸出入埠， 硬體中斷或直接記憶體存取通道。（既然 AMD53C974 是個 PCI 設備，其實也不需要作這些設定。） 取而代之的是用來傳達介面卡與設備間傳輸模式與傳輸率的參數。 這最好用範例來解釋：

```
AM53C974=7,2,8,15
```

這會被解釋成這樣：‘對於 SCSI-ID 7 的控制器與 SCSI-ID 2 的設備間之通訊， 使用最大 8MHz 之同步模式傳輸率。應協商 15 位元組的偏移值。’ 詳細資料可在 `linux/drivers/scsi/README.AM53C974` 檔案裡找到。

4.2.7 BusLogic SCSI 介面卡與 v1.2 的核心 (‘buslogic=’)

在舊版的核心中， buslogic 驅動程式只接受一個參數，就是輸出入位址。 它能接受下列值：0x130，0x140，0x230，0x234，0x330，0x334。

4.2.8 BusLogic SCSI 介面卡與 v2.x 的核心 (‘BusLogic=’)

在 v2.x 核心中，BusLogic 驅動程式接受許多參數。 （注意上面的大小寫；是大寫的 B 以及 L!!! ）。 下面的細節是直接從 v2.0 核心中的 Leonard N. Zubkoff’s 驅動程式拿過來的。

這個 BusLogic 驅動程式的核心命令列是由 "BusLogic" 驅動程式識別字， 加上各個以逗號格開的選用性整數序列，後面跟著各個選用性的字串序列所組成。 每行命令列都用於一張 BusLogic 卡， 在使用多張 BusLogic 卡的系統上可以使用多個命令列來設定。

第一個整數指定介面卡所在的 I/O 位址。如果不指定則預設值爲零， 表示把這個參數設爲預設偵測順序中找到的第一張 BusLogic 卡。 如果在命令列上指定了任何的 I/O 位址參數，預設偵測順序就會被忽略。

第二個整數指定用於支援佇列運作(Tagged Queuing)的目標設備之佇列深度。 佇列深度是容許同時送出的 SCSI 指令數。如果沒有指定則預設值爲零， 表示使用根據介面卡的總佇列深度及數目，種類，速度， 以及偵測目標設備的能力所自動判斷的值。 對於需要使用 ISA 錯誤緩衝區(Bounce Buffers)的介面卡， 佇列深度自動設爲 BusLogic_TaggedQueueDepth_BB 以避免 DMA 錯誤緩衝記憶體的過載。 不支援佇列運作的目標設備使用 BusLogic_UntaggedQueueDepth 定義的佇列深度。

第三個整數指定匯流排停置(Settle)時間，以秒爲單位。 這乃是重置介面卡使 SCSI 匯流排重置到發出 SCSI 指令之間的時間。 如果沒有指定則預設值爲零，表示使用 BusLogic_DefaultBusSettleTime 的值。

第四個整數指定區域選項。如果沒有指定則預設值爲零。 注意，區域選項只用於特定的介面卡。

第五個整數指定全域選項。如果沒有指定，預設值爲零。 注意，全域選項應用於所有的介面卡。

字串選項用來提供對佇列運作，錯誤回復，以及介面卡偵測之控制。

佇列運作的指定以 "TQ:" 開頭， 可以明確地指定是否允許有此支援的目標設備使用佇列運作。 下面是可用的指定選項：

TQ:Default – 佇列運作的允許與否以 BusLogic 卡的韌體版本以及佇列深度的值是否允許多重指令作爲判斷基礎。

TQ:Enable – 對所有接在這張卡上的目標設備打開佇列運作，忽略介面卡韌體版本所加諸的限制。

TQ:Disable – 對所有接在這張卡上的目標設備關閉佇列運作。

TQ:<每個目標的設定> – 各別控置對每個目標設備的佇列運作。 <每個目標的設定>是由 "Y", "N", and "X" 字元組成的串列。 "Y" 打開佇列運作，"N" 關閉佇列運作，而 "X" 接受以韌體版本爲基礎的預設值。 第一個字元代表目標設備 0，第二個代表設備 1，依此類推； 如果 "Y", "N" and "X" 字元串列並未涵蓋所有的目標設備， 未指定的字元將被假設爲 "X"。

注意，明確地指定要使用佇列運作可能會出問題； 這主要是方便用來關閉對某些沒能正確實作此功能之目標設備的佇列運作。

錯誤回復策略的指定以 "ER:" 開頭，可以明確地指定因 SCSI 指令失敗， 無法成功完成而呼叫 ResetCommand 時執行的錯誤回復動作。 下面是可用的指定選項。

ER:Default – 錯誤回復依 SCSI 子系統的建議選擇強制重置(Hard Reset)與匯流排設備重置其中之一。

ER:HardReset – 錯誤回復將會啓始介面卡的強制重置，這也會引起 SCSI 匯流排的重置。

ER:BusDeviceReset – 錯誤回復將送出匯流排設備重置訊息給引起錯誤的目標設備。

果該目標設備再次造成錯誤回復而且送出匯流排設備重置訊息之後該目標設備沒有一次成功完成 SCSI 指令的話， 則嘗試進行強置重置。

ER:None – 錯誤回復將被凍結。 這個選項應該只用於 SCSI 匯流排重置或匯流排設備重置會使目標設備無法回復的情況。

ER:<每個目標的設定> – 錯誤回復將對每個目標設備進行各別控制。 <每個目標的設定>是由 "D", "H", "B", and "N" 字元所組成的串列。 "D" 選擇預設值，"H" 選擇強置重置，"B" 選擇匯流排設備重置，而 "N"選擇不重置。 第一個字元代表目標設備 0，第二個代表設備 1，依此類推； 如果 "D", "H" , "B" and "N" 字元串列並未涵蓋所有的目標設備， 未指定的字元將被假設爲 "X"。

介面卡偵測的指定包含下列字串：

NoProbe – 不執行任何一種偵測，因此不會偵測到 BusLogic 介面卡。

NoProbeISA – 不偵測標準 ISA I/O 位址，因此只會偵測到 PCI 介面卡。

NoSortPCI – 依 PCI BIOS 所提供的順序列舉 PCI 介面卡， 忽略任何 AutoSCSI "Use Bus And Device # For PCI Scanning Seq." 選項的設定。

4.2.9 EATA SCSI 介面卡 (‘eata=’)

新的 v2.0 核心之 EATA 驅動程式能接受一個啟動參數指定所要偵測的輸出入位址。 格式爲：

```
eata=iobase1[,iobase2][,iobase3]...[,iobaseN]
```

此驅動程式會依所列順序偵測這些位址。

4.2.10 Future Domain TMC-8xx, TMC-950 (‘tmc8xx=’)

偵測這些 SCSI 卡的程式碼尋找從卡上安裝進來的基本輸出入系統， 如果沒有，就找不到你的卡。 或者，如果 BIOS 的簽章(signature)字串不被認可那麼也會找不到。 不管是那一種情況，你都得使用這種型式的啟動參數：

```
        tmc8xx=mem_base,irq
```

`mem_base` 的值是界面卡使用的記憶體對映(memory mapped)輸出入區域的值。 此值通常會是下列其中之一： `0xc8000`, `0xca000`, `0xcc000`, `0xce000`, `0xdc000`, `0xde000`。

4.2.11 Future Domain TMC-16xx, TMC-3260, AHA-2920 (‘fdomain=’)

這個驅動程式根據一份已知基本輸出入系統唯讀記憶體簽章列表偵測這些介面卡。 完整的列表請參見 `linux/drivers/scsi/fdomain.c`， 該檔前面有許多資訊。 如果此驅動程式不認得你的基本輸出入系統，你可以使用這種格式變更：

```
        fdomain=iobase,irq[,scsi_id]
```

4.2.12 IOMEGA Parallel Port / ZIP drive ('ppa=')

這個驅動程式用於 IOMEGA ZIP 磁碟附的 IOMEGA 並列埠 SCSI 卡。 它也可以配合原先的 IOMEGA PPA3 設備使用。 此驅動程式的啓動參數格式爲：

```
ppa=iobase,speed_high,speed_low,nybble
```

除了 iobase 之外其餘皆爲選用值。如果你想改變這三個選用參數中的任何一個， 建議你讀讀 `linux/drivers/scsi/README.ppa` 中有關它們控制些什麼的細節。

4.2.13 NCR5380 based controllers ('ncr5380=')

跟你的介面卡有關， 5380 的型式可以是輸出入對應或記憶體對應。 （低於 0x400 的位址意謂著輸出入對應， 但 PCI 以及 EISA 硬體使用 0x3ff 以上的輸出入位址。）不管是那一種情況， 你都是指定位址，硬體中斷以及直接記憶體存取通道的值。 輸出入對應介面卡的範例像是： `ncr5380=0x350,5,3`。 如果該卡不使用中斷，那麼 255(`0xff`)這個硬體中斷值會關閉中斷的使用。 254 這個應體中斷值表自動偵測。 更細節的資訊可以在 `linux/drivers/scsi/README.g_NCR5380` 這個檔案裡找到。

4.2.14 NCR53c400 based controllers ('ncr53c400=')

標準 53c400 的支援是由與上述標準 5380 之支援相同的驅動程式所提供。 啓動參數與上述完全相同， 除了 53c400 不使用直接記憶體存取之外。

4.2.15 NCR53c406a based controllers ('ncr53c406a=')

此驅動程式使用這種格式的啓動參數：

```
ncr53c406a=PORTBASE,IRQ,FASTPIO
```

其中 IRQ 以及 FASTPIO 參數爲選用項。中斷值零關閉中斷之使用。 使用一與 FASTPIO 參數啓用 `insl` 以及 `outsl` 指令以取代一個位元組的 `inb` 以及 `outb` 指令。 此驅動程式也可以由編譯時期選項設定使用直接記憶體存取。

4.2.16 Pro Audio Spectrum ('pas16=')

PAS16 使用 NCR5380 SCSI 晶片，新款還支援無跳接配置。啓動參數的格式爲：

```
pas16=iobase,irq
```

唯一的不同是你可以指定 255 這個硬體中斷值， 這會告訴驅動程式不要使用中斷，然效能會降低。 `iobase` 通常是 `0x388`。

4.2.17 Seagate ST-0x ('st0x=')

偵測這些 SCSI 卡的程式碼尋找從卡上安裝進來的基本輸出入系統， 如果沒有，就找不到你的卡。或者，如果 BIOS 的簽章字串不被認可那麼也會找不到。 不管是那一種情況，你都得使用這種型式的啓動參數：

```
st0x=mem_base,irq
```

mem_base 的值是界面卡使用的記憶體對映(memory mapped)輸出入區域的值。 此值通常會是下列其中之一： 0xc8000, 0xca000, 0xcc000, 0xce000, 0xdc000, 0xde000。

4.2.18 Trantor T128 ('t128=')

這些介面卡也是以 NCR5380 晶片爲基礎，接受下列選項：

```
t128=mem_base,irq
```

The valid values for mem_base are as follows: 0xcc000, 0xc8000, 0xdc000, 0xd8000. 可用於 mem_base 的值如下： 0xcc000, 0xc8000, 0xdc000, 0xd8000。

4.2.19 Ultrastor SCSI cards ('u14-34f=')

注意，此卡有兩個互相獨立的驅動程式，名爲 CONFIG_SCSI_U14_34F 者使用 u14-34f.c 而 CONFIG_SCSI_ULTRASTORE 使用 ultrastor.c。 u14-34f 這一個（新的 v2.0 核心用它）接受一個格式如下的啓動參數：

```
u14-34f=iobase1[,iobase2][,iobase3]...[,iobaseN]
```

此驅動程式會依所列順序偵測這些位址。

4.2.20 Western Digital WD7000 cards ('wd7000=')

這個驅動程式尋找它所知道的基本輸出入系統唯讀記憶體字串來偵測 wd7000 並且知道數個標準的配置設定。 如果它沒能用正確的值動起來，或不認得你的基本輸出入系統版本， 那麼你可以使用這種格式的啓動參數。

```
wd7000=irq,dma,iobase
```

4.3 不接受啓動參數的 SCSI 卡

目前爲止，下列 SCSI 介面卡不使用任何啓動時期參數。在某些情況下， 你可以直接編輯驅動程式本身把值寫死(*hard-wire*)進去，如果眞的必要的話。

```
Adaptec aha1740 (EISA probing),
NCR53c7xx,8xx (PCI, both drivers)
Qlogic Fast (0x230, 0x330)
Qlogic ISP (PCI)
```

5 硬碟機

這一節列出所有與標準 MFM/RLL, ST-506, XT, 以及 IDE 磁碟機相關的啓動參數。 注意，IDE 以及先前的 ST-506 硬碟驅動程式都接受 ‘hd=’ 選項。

5.1 IDE 磁碟光碟驅動程式參數

IDE 驅動程式接受幾個參數， 涵蓋範圍從磁碟規格(geometry)到先進或不良之控制器的支援。 底下是所有可用之啓動參數的摘要。 要知道完整的細節， 你眞的應該去查閱在 `linux/Documentation` 目錄下的 `ide.txt` 檔， 這個摘要是從中摘錄出來的。

```
"hdx="  從 "a" 到 "h" 所有的 "x" 都承認，像是 "hdc"。
"idex=" 從 "0" 到 "3" 所有的 "x" 都承認，像是 "ide1"。

"hdx=noprobe"          : 也許的確有個磁碟在那兒，但不要去偵測
"hdx=none"             : 這裡沒有磁碟，忽略 cmos 且不要去偵測
"hdx=nowerr"           : 忽略這個磁碟的 WRERR_STAT 位元
"hdx=cdrom"            : 這裡有個磁碟，而且是個光碟機
"hdx=cyl,head,sect"    : 這裡有個磁碟，並指定磁碟規格
"hdx=autotune"         : 驅動程式將試著把介面速度調到支援之
                              最快可程式化輸出入(PIO) 模式
                             ，此磁碟能接受才行。並非所有
                              晶片組種類都有完整支援，並且
                              很可能使舊的奇怪的 IDE  磁
                              碟出問題。
"idex=noprobe"         : 不要嘗試存取使用這個介面
"idex=base"            : 在指定的位址偵測介面，其中 "base"
                              通常是 0x1f0  或是 0x170  且
                              假設 "ctl"  是 "base"+0x206
"idex=base,ctl"        : 指定 base 以及 ctl
"idex=base,ctl,irq"    : 指定 base, ctl, 以及 irq  數值
"idex=autotune"        : 驅動程式將試著把介面速度調到支援之
                              最快可程式化輸出入(PIO) 模式
                             ，此磁碟能接受才行。並非所有
                              晶片組種類都有完整支援，並且
                              很可能使舊的奇怪的 IDE  磁
                              碟出問題。
"idex=noautotune"      : 驅動程式將不會試著調整介面速度。除
                              了 cmd640 之外，這是大部份晶
                              片組的預設值。
"idex=serialize"       : 不重疊對(overlap) idex 及 ide(x^1)
                              的操作
```

下列參數只允許用於 ide0， 且預設的 base, ctl 以及 ports 不能修改。

```
"ide0=dtc2278"          : 偵測支援 DTC2278 介面
"ide0=ht6560b"          : 偵測支援 HT6560B 介面
"ide0=cmd640_vlb"       : 使用 CMD640 晶片的 VLB  介面卡
                          *必須*加入這個參數（PCI 的則
                          不必 -- 自動偵測）
"ide0=qd6580"           : 偵測支援 qd580 介面
"ide0=ali14xx"          : 偵測支援 ali14xx 晶片組 (ALI M1439/M1445)
"ide0=umc8672"          : 偵測支援 umc8672 晶片組
```

使用任何其它的東東都會被拒絕並得到一個 "BAD OPTION" 訊息。

5.2 標準 ST-506 磁碟驅動程式選項(‘hd=’)

標準磁碟驅動程式可以接受磁碟邏輯規格參數，這與 IDE 驅動程式相似。注意，無論如何它只希望接受三個值 (C/H/S) – 過多或過少它都會默默地忽略掉。同時它只接受 ‘hd=’ 參數，像 ‘hda’, ‘hdb=’ 這類的在此都不被允許。其格式如下：

```
hd=cyls,heads,sects
```

如果裝有兩個磁碟，就使用第二顆磁碟的規格重覆上述指令。

5.3 XT Disk Driver Options (‘xd=’)

如果你不幸到得要使用這些舊的、8 位元、以驚人的 125kb/s 速度搬移資料的介面卡那麼這節是你的獨門解藥。偵測這些卡的程式碼尋找從卡上安裝進來的基本輸出入系統，如果沒有，就找不到你的卡。或者，如果 BIOS 的簽章字串不被認可那麼也會找不到。不管是那一種情況，你都得使用這種型式的啓動參數：

```
xd=type,irq,iobase,dma_chan
```

其中 `type` 值指示該介面卡的製造商，如下：0=generic; 1=DTC; 2,3,4=Western Digital, 5,6,7=Seagate; 8=OMTI。同一家製造商種類之間的不同僅止於用來偵測的基本輸出入系統字串，這個東東若指定 type 就會不使用。

這些值 `xd_setup()` 函式不會去檢查，並且假設你輸入所有共四個值。別讓它失望。這兒有個範例使用 WD1002 控制器並關閉拔掉基本輸出入系統，使用‘預設的’XT 控制器參數：

```
xd=2,5,0x320,3
```

6 光碟機 (Non-SCSI/ATAPI/IDE)

這一節列出屬於光碟設備所有可能的啓動參數。注意，這不包含 SCSI 或是 IDE/ATAPI 光碟機。這些光碟機請參閱其它適當的章節。

注意，大部份光碟機都有些文件檔值得一讀，而且它們都放在一個很容易取得的地方：`linux/Documentation/cdrom`。

6.1 Aztech 介面 (‘aztcd=’)

這種介面卡的語法是：

```
aztcd=iobase[,magic_number]
```

如果你將 magic_number 設為 0x79 那麼此驅動程式不管三七二十一都會試著在未知的韌體版本上跑。所有其它的值都會被忽略。

6.2 CDU-31A 以及 CDU-33A Sony 介面 (‘cdu31a=’)

此光碟機介面出現在部份 Pro Audio Spectrum 音效卡以及其它 Sony 出產的介面卡上。語法如下：

```
cdu31a=iobase,[irq[,is_pas_card]]
```

指定零作為 IRQ 值告訴此驅動程式不支援硬體中斷（某些 PAS 的卡）。 如果你的卡支援中斷，應該使用之以降低此驅動程式的中央處理器使用率。

如果使用 Pro Audio Spectrum 的介面卡則 ‘is_pas_card’ 欄應該輸入 ‘PAS’， 若否就不應該使用。

6.3 CDU-535 Sony 介面 (‘sonycd535=’)

這種光碟機介面卡的語法是：

```
sonycd535=iobase[,irq]
```

如果你想指定 IRQ 值，輸出入位址可以填入零來佔位置(placeholder)。

6.4 GoldStar 介面 (‘gscd=’)

這種光碟機介面卡的語法是：

```
gscd=iobase
```

6.5 ISP16 介面 (‘isp16=’)

這種光碟機介面卡的語法是：

```
isp16=[port[,irq[,dma]]][[,]drive_type]
```

使用零作為 irq or dma 表示不使用該功能。 可用於 drive_type 的值是 noisp16, Sanyo, Pansonic, Sony, and Mitsumi。使用 noisp16 會徹底關掉這個驅動程式。

6.6 Mitsumi 標準介面 ('mcd=')

這種光碟機介面卡的語法是：

```
mcd=iobase,[irq[,wait_value]]
```

其中 wait_value 是作爲內部的逾時設定，用在那些不聽話的機子上， 這個功能是否有實作是依編譯時期的定義值而定。

6.7 Mitsumi XA/MultiSession 介面 ('mcdx=')

目前爲止，此‘實驗性’的驅動程式有個設定函式， 但並未實作任何參數(as of 1.3.15)。它用於與上述驅動程式相同的硬體， 但此驅動程式具有更多功能。

6.8 Optics Storage 介面 ('optcd=')

這種介面卡的語法是：

```
optcd=iobase
```

6.9 Phillips CM206 介面 ('cm206=')

這種介面卡的語法是：

```
cm206=[iobase][,irq]
```

這個驅動程式假設介於 3 到 11 之間數字是 IRQ 值， 而 0x300 到 0x370 之間的數字則是輸出入埠，所以你可以只指定一個值或二個都指定，順序隨意。 它也接受 'cm206=auto' 以打開自動偵測。

6.10 Sanyo 介面 ('sjcd=')

這種介面卡的語法是：

```
sjcd=iobase[,irq[,dma_channel]]
```

6.11 SoundBlaster Pro 介面 ('sbpcd=')

這種介面卡的語法是：

```
sbpcd=iobase,type
```

其中的 type 是下列之字串一（大小寫有差）：'SoundBlaster', 'LaserMate', or 'SPEA'。 輸出入位址是光碟機的輸出入位址，不是該卡音效部份的輸出入位址。

7 其它硬體設備

其它任何不適合放在上面分類中的設備都堆在這裡。

7.1 乙太網路設備 (‘ether=’)

不同的驅動程式使用不同的參數， 但它們至少都共用 IRQ, I/O 埠位址值以及名稱。最標準的格式看起來像：

```
ether=irq,iobase[,param_1[,param_2,...param_8]]],name
```

第一個不是數字的參數被當作名稱。param_n 的值（如果配得起來）對每張不同的介面卡驅動程式通常都代表不同的意義。 典型的 param_n 值是用來指定像共享記憶體位址，介面的選擇， DMA 通道等等諸如此類的東東。

此參數最常見的用法是在預設值僅偵測到一張網路卡的時候強制偵測第二張乙太網路卡。 可以用這個簡單的指令達成：

```
        ether=0,0,eth1
```

注意，上述範例中 IRQ 與 I/O 位址零告訴驅動程式自動去偵測。

模組使用者者請注意： 如果驅動程式作爲執行期間可載入模組（不編譯到核心裡） 使用則上述指令不會強制偵測第二張介面卡。 大部份的 Linux 發行套件都使用一個基本的核心配合許許多多做成模組的驅動程式。 此 ether= 參數只對直接編譯在核心裡的驅動程式有效。

在乙太網路說明(Ethernet-HowTo)中對使用多張網路卡以及 param_n 值的使用場合，網路卡驅動程式對這些值的實作等等有更完整、更廣泛的文件。 有興趣的讀者應該參考該文件中有關自己網路卡的章節。 *Ethernet-HowTo* <http://sunsite.unc.edu/mdw/HOWTO/Ethernet-HOWTO.html>

7.2 軟碟驅動程式 (‘floppy=’)

軟碟驅動程式有許多選項， 全部都列在 linux/drivers/block 裡的 README.fd 中。 這是直接從該檔取得的資訊。

floppy=mask,allowed_drive_mask

設定允許使用之磁碟機位元遮罩爲 mask。 依預設，只允許每個軟碟控制器的第零及第一個單元。 這樣做是因爲有些非標準的硬體（華碩的 PCI 主機板） 在存取第二或第三個單元時會當掉鍵盤。此選項差不多已經被 cmos 選項給取代了。

floppy=all_drives

設定允許使用之磁碟機位元遮罩爲所有磁碟機。 如果你在軟碟控制器上接了兩個以上的磁碟機的話可以使用。

floppy=asus_pci

設定位元遮罩爲只允許第零及第一個單元。（預設值）

floppy=daring

告訴軟碟驅動程式你有個行爲良好的軟碟控制器。 這樣運作上會比較有效率、比較平順，但有些控制器可能無法使用。 這可以加速某些部份的運作。

floppy=0,daring

告訴軟碟驅動程式你的軟碟控制器得小心看管。

floppy=one_fdc

告訴軟碟驅動程式你只有一個軟碟控制器。（預設值）

floppy=two_fdc *or* floppy=address,two_fdc

告訴軟碟驅動程式你有兩個軟碟控制器。 第二個軟碟控制器假設位於 address 指定的地方。 如果沒有給 address，假設爲 0x370。

floppy=thinkpad

告訴軟碟驅動程式你有一台 Thinkpad。 Thinkpads 的磁碟變更偵測線路(disk change line)之使用與一般相反。

floppy=0,thinkpad

告訴軟碟驅動程式你沒有 Thinkpad。

floppy=drive,type,cmos

設定 `drive` 的 cmos 類型爲 `type`。 條件是，這個磁碟機在位元遮罩中必須是可用的。 如果你有兩個以上的磁碟機（實體 cmos 只能指定兩個） 或是你的基本輸出入系統使用非標準的 CMOS 類型那這就有用了。 把前兩個磁碟機的 cmos 設爲 0（預設值） 會使得軟碟驅動程式爲這些磁碟機讀取實體 cmos。

floppy=unexpected_interrupts

接收到非預期之中斷時印出警告訊息（預設）

floppy=no_unexpected_interrupts *or* floppy=L40SX

接收到非預期之中斷時不印出警告訊息。 IBM L40SX 膝上型電腦在某些顯示模式下需要它。 （這看來是影像與軟碟之間的某種相互關係。 此非預期之中斷只會影響效能，可以安全地忽略。）

7.3 音效驅動程式 (‘sound=’)

音效驅動程式也可以接受啓動參數以變更編譯時的值。 不過並不建議這樣做，因爲這樣會更複雜。 相關說明放在（曾經？）Readme.Linux 中， 位於 `linux/drivers/sound` 裡。它接受這種型式的參數：

```
sound=device1[,device2[,device3...[,device11]]]
```

其中每個 `deviceN` 值的格式爲下列的 `0xTaaaId` 而每個位元組的使用如下：

T - 設備種類: 1=FM, 2=SB, 3=PAS, 4=GUS, 5=MPU401, 6=SB16, 7=SB16-MPU401

aaa - 十六進位的輸出入位址。

I - 十六進位之中斷線 (i.e 10=a, 11=b, ...)

d - 記憶體直接存取通道。

如你所見，這相當雜亂，建議你最好把自己設備的值編譯進去。 使用啓動參數 ‘sound=0’ 將會玩全關閉音效驅動程式。

7.4 匯流排滑鼠驅動程式 ('bmouse=')

匯流排滑鼠驅動程式只接受一個參數，就是硬體所使用的 IRQ 值。

7.5 微軟匯流排滑鼠驅動程式 ('msmouse=')

微軟滑鼠驅動程式只接受一個參數，就是硬體所使用的 IRQ 值。

7.6 印表機驅動程式 ('lp=')

在高於 1.3.75 以上的核心上你可以告訴印表機驅動程式使用那個埠以及不使用那個埠。如果你不想讓印表機驅動程式拿到所有可用的並列埠那麼後者很方便，這樣其它的驅動程式(e.g. PLIP, PPA)就可以使用它們。

這個參數的格式是多個 i/o, IRQ 的組合。例如，`lp=0x3bc,0,0x378,7` 會以不需 IRQ(polling) 的模式使用位於 0x3bc 的埠，並且對位於 0x378 的埠使用 IRQ 7。位於 0x278 的埠（如果有的話）將不會被偵測到，因爲自動偵測只用於沒有 'lp=' 參數的情況下。可以使用 `lp=0` 關閉整個印表機驅動程式。

7.7 ICN 整體服務數位網路(ISDN)驅動程式 ('icn=')

這個整體服務數位網路驅動參數的啓動參數格式爲：

```
icn=iobase,membase,icn_id1,icn_id2
```

其中 `iobase` 是該卡的輸出入埠位址，`membase` 是該卡的共享記憶體基底位址，而那兩個 `icn_id` 是唯一識別的 ASCII 辨識字串。

7.8 PCBIT ISDN 驅動程式 ('pcbit=')

這個啓動參數格式使用一對整數參數：

```
pcbit=membase1,irq1[,membase2,irq2]
```

其中 `membaseN` 是第 N 張卡的共享記憶體基底，而 `irqN` 是第 N 張卡的中斷設定。預設值是 IRQ 5 以及 membase `0xD000`。

7.9 Teles ISDN 驅動程式 ('teles=')

這個 ISDN 驅動程式的啓動參數格式爲：

```
teles=iobase,irq,membase,protocol,teles_id
```

其中 `iobase` 是該卡的 i/o 埠位址，`membase` 是該卡的共享記憶體基底位址，`irq` 是該卡使用的中斷，而 `teles_id` 是唯一識別的 ASCII 辨識字串。

7.10 DigiBoard 驅動程式 (‘digi=’)

DigiBoard 驅動程式接受由六個逗號隔開之的辨識字串或整數。 六個值依序爲：

打開關閉該卡
卡的種類： PC/Xi(0), PC/Xe(1), PC/Xeve(2), PC/Xem(3)
打開關閉其它針腳排列(alternate pin arrangement)
該卡上的埠數
該卡設定的輸出入埠（如果以字串指定則爲十六進位）
記憶窗的基底（如果以字串指定則爲十六進位）

正確的啓動參數之範例（字串以及整數格式）如：

```
digi=E,PC/Xi,D,16,200,D0000
digi=1,0,0,16,512,851968
```

注意，這個驅動程式若無 digi= 啓動參數其預設輸出入爲 0x200， 共享記憶體基底爲 0xD0000。不執行自動偵測。 更進一步的細節可以在 linux/Documentation/digiboard.txt 檔案裡找到。

7.11 RISCom/8 多埠串列卡驅動程式 (‘riscom8=’)

Up to four boards can be supported by supplying four unique i/o port values for each individual board installed. Other details can be found in the file linux/Documentation/riscom8.txt.

可以支援達四張卡，須提供四個唯一的輸出入埠給每張卡。 其餘細節可以在 linux/Documentation/riscom8.txt 檔案裡找到。

7.12 Baycom 串列並列無線數據機 (‘baycom=’)

這些設備的啓動參數格式如下。

```
baycom=modem,io,irq,options[,modem,io,irq,options]
```

使用 modem=1 表示你有 ser12 設備，modem=2 表示你有 par96 設備。 使用 options=0 表示使用硬體 DCD，options=1 則表示使用軟體 DCD。 io 以及 irq 跟往常一樣是輸出入埠以及中斷設定。 在 /linux/drivers/char/ 目錄下的 README.baycom 檔案裡有更多的細節。

8 結束

如果你發現這份文件裡有任何醒目的打字錯誤，或過時的資訊，請讓 我知道。這很容易就會看漏了。

謝謝，

Paul Gortmaker, gpg109@rsphy1.anu.edu.au