

发布我的Linux 0.11 实验室（带有gcc4 下可编译的源码）

<http://www.quinnli.com/>

[下载](#)

简介

我发布的这个东东主要包含一个可以在 gcc 4.1 下可编译的 linux-0.11 的修改版的源码。当前的 Linux 发型版大部分已经采用 gcc 4.0 以上的版本和 linux 2.6 的内核，而赵博士 (www.oldlinux.org) 提供的可以在 RH9 下面编译的源码只能工作于 gcc 版本 3.x。

我在我的 Debian Sid 系统上对这个东东做了基本的测试，gcc 4.1 是我系统的默认编译器。任何一个比较新的 Linux 发型版，包括 Fedora, Ubuntu 等，应该都没有问题。如果你的系统上安装的是其他版本的 gcc，可以参考下面“使用其他 gcc 编译”小节。

如果你的 Linux 提供的默认编译器的仍然是 gcc 3.x，或者你打算另外安装一个 gcc 3.x 仅仅为了满足能编译源码的要求，你可能就用不到这个东东了。但是，除了内核源码之外，我还提供了一整套 linux-0.11 的实验环境，用于在 linux 环境下学习并且实验 linux 0.11 的源代码。因为我认为要想真正的学习 Linux 内核的精髓，还是得老老实实在 Linux 环境中进行学习。而且在 Windows 下面，几乎就不可能编译 Linux 内核的源码。

如果你真的很想在 Windows 平台学习 Linux 也没有关系。我还做了另外一个类似的东东可以在 Windows 平台直接学习 Linux 0.11 的源码。只需要下载下面的链接并且解压到系统的某个地方就可以用：

http://www.quinnli.com/upload/linux-0.11-lab_060616_004102.exe

无论是 Linux 用户也好，Windows 用户也好，如果你也是新学 Linux，不太清楚怎么样修改并且编译内核源码，然后重启进入你修改后的内核这些基本操作，可以看看我做的一个 Flash 教程：

<http://www.quinnli.com/upload/linux-0.11.swf>

这个东东很大部门都是直接用了 www.oldlinux.org 提供的资源做成的。我只是根据自己的需求，综合了一些资源，做了个简单的定制。非常感谢赵博士共享这些资料，并且希望我的东东能对大家有点用处。谢谢支持。

你可以访问[这个链接](#)报bug和给这个东东提一些建议。我其实也是刚学Linux内核的菜鸟。

运行前提

你的系统中需要预先安装 mtools 和 bochs。如果你用基于 Debian 的发型版，可以运行：

```
apt-get install bochs bochs-x mtools
```

如果你的系统是基于 RPM 的，可以试试：

```
yum install bochs mtools
```

或者直接运行 bochs 和 mdir 这两个命令，如果都有就说明你的系统中已经安装了这两款软件。

为了编译内核源码，当然你的系统中还必须装有 gcc 和 make 等基本软件，如果是 Debian，可以运行：

```
apt-get install build-essentials bin86
```

安装

解压这个文件都可以了：

```
tar xjf linux-0.11-lab_060616_005627.tar.bz2
```

然后在本目录下应该会出现一个名为 linux-0.11 的新目录。

工作流程

使用这个东东的工作过程很简单，首先编译内核，然后安装内核，最后在 bochs 中运行内核。

假设你已经进入 linux-0.11 这个目录，首先运行：

```
cd source/linux  
make clean  
make  
cd ../../
```

将在 source/linux 目录下生成新的内核文件 Image。然后运行：

```
./mcopy_kernel source/linux/Image bootimage-new
```

可以将 Image 文件拷贝到一个装了 grub 的启动软盘上。这个命令将在磁盘的 /boot 目录下建立 bootimage-new 这个文件，值得指出的是，如果用这个命令，所有的都会被拷贝到软盘的 /boot 目录下。

然后你可以运行

```
./edit_menu.lst
```

来编辑 grub 的配置文件，这样启动是就可以在 grub 菜单中显示新内核的启动项。配置文件语法很简单。

比如说你可以在 menu.lst 最后加上以下四行：

```
title           Linux 0.11 compiled by gcc 4.1
root            (fd0)
kernel          /boot/bootimage-new
boot
```

最后可以运行：

```
./linux-0.11
```

来启动 bochs 虚拟机。如果没有别的问题，你应该可以看到 grub 启动菜单，这时候选择你刚刚创建的菜单项就可以进入新的内核。

另外，如果你不想在自身的Linux环境下编译源码也可以，因为你可以在虚拟机中的 linux-0.11 环境中编译linux-0.11 的源码。如果你决定这样做，可以看我的作的[flash教程](#)。

目录结构

source/

linux/

这个目录下有我修改的源码

linux-0.11-040327-rh9.tar.gz

这个是 www.oldlinux.org 提供的在 RH9 下编译的原始代码

linux-0.11-040327-rh9.diff.gz

这是 www.oldlinux.org 提供的为了能在 RH9 下编译提供的差异文件

linux-0.11-060617-gcc4-diff.gz

这是我基于rh9 版本作出的修改，基本上都是为了编译通过作出的很简单的修改。但是有一处修改并不那么直观，可以参考我写的[这篇文章](#)

conf/

这个目录下有配置文件

images/

这个目录下包含虚拟机需要的软盘和硬盘镜像文件。硬盘镜像文件是 www.oldlinux.org 提供的包含完整 Linux 开发环境的硬盘镜像。还有一张 grub 启动软盘，你可以将新的内核文件拷贝到这张软盘中。

bak/

每次运行 `./edit_menu.lst` 和 `./mcopy_kernel` 都会备份当前的 grub 启动盘到 bak 目录中。必要的时候可以从这个目录下恢复。

使用其他版本的 gcc 编译

使用其他版本的 gcc 来编译也是可以的，gcc-4.0 应该可以编译。而且如果你用的系统非常非常老，还在使用 gcc-2.95，经过我的测试，也可以支持。

但如果你的默认版本的 gcc 是 3.x，其实你不需要使用我修改的版本，使用最初赵炯博士提供的 rh9 版本就可以。这个包里也包含了这个文件

如果你实在是想用 gcc-3.3 编译我的版本，也行，只不过需要首先将源码目录中的 ``mm/Makefile'` 替换为 ``mm/Makefile-gcc3.x'`。原始的 ``mm/Makefile'` 也有其备份 ``mm/Makefile-gcc-4.1'`，必要时可以恢复。一定记得覆盖，不然在编译的时候不会出现任何问题，但是编译出来的内核镜像将无法启动。

已知问题

1. 我的内核无法挂接软盘上的 rootdisk

是的，如果你和我一样，使用当前的 2.2.6 版本的 bochs，而且是直接把编译出来的 Image 文件做为 a 盘启动，就有可能出现问题。如果你肯定的确在 b 盘放入了 rootimage 文件的话，我也不知道为什么会这样。

但是如果使用 grub 来引导信的 Image 文件，就完全不会出现这个问题。我的猜测是 grub 在引导 Image 之前对软驱做了一些必要的初始化操作，而 Linux 0.11 没有做。

2. 我的内核启动是只能用软盘 b 做为根分区

是的，这是默认的启动设备。你可以使用 `rdev` 命令改变一个磁盘镜像的启动设备（必须 `su` 到 `root` 来执行这个操作）

```
rdev source/linux/Image /dev/hda1
```

修改完之后使用 `./mcopy_kernel` 将新的镜像复制到 grub 盘上。

（令我非常惊奇的是 `rdev` 竟然可以修改 linux 0.11 这么古老的版本。这兼容性太好了）

3. 我的内核镜像虽然已经启动了，但是好多命令不能正确执行，总是出错说 ``not owner``

是的。如果你已经改用硬盘 root image, 可以简单解决这个文件, 首先备份一下 /bin/sh, 然后将/bin/sh0 覆盖/bin/sh, 重启系统之后就应该正常了。

至于为什么, 可以进一步参考一下论坛上的[这篇文章](#)

Technorati Tags: [linux](#), [内核](#), [kernel](#), [开发](#)

[Add comment](#) *Jun 18, '06, 04:00am quinn.liqin*

[Release My Linux 0.11 Lab with modified gcc4 compatible source code](#)

[download it](#)

Introduction

This package contains a modified version of linux-0.11 to be compiled under modern linux distribution with gcc-4.1 and linux kernel 2.6. Because original modified version provided by Dr. Zhao from www.ouldlinux.org will only work under gcc-3.x

It is tested in my Debian Sid box with gcc-4.1 as the default gcc compiler. Any other up-to-date distributions like FC, Ubuntu should also work. Other gcc version should also work, please see section [`use other version of gcc`](#)

But if your default compiler is gcc-3.x or you choose to install gcc-3.x only to satisfy the code, you might not need this package. But other than the source code, I also provide a complete linux-0.11 laboratory to be used under linux, NOT windows. Because I believe no one can actually learn any thing about linux without really work with linux. And it's nearly impossible to compile linux source code under windows.

If you use windows and really want to learn linux-0.11 under windows. That's ok too, I provided another package for windows users, all you have to do is to download and extract the files into you system and run. Here:

http://www.quinnli.com/upload/linux-0.11-lab_060616_004102.exe

For both linux and windows users, if you are new to linux, and don't know how to actually changed the source code of linux-0.11 and reboot to see the new feature, I've made a flash tutorial to demonstrate some essential operations:

<http://www.quinnli.com/upload/linux-0.11.swf>

My work is heavily based on many materials provided by www.ouldlinux.org. What I have done for this package is only some customization to meet my own requirement to learn linux. Many many thanks Dr. Zhao for his dedicated work, and hope mine would be useful to you too. Thanks.

You can go to [this page](#) to report bugs and comments on this package. I'm a linux kernel rookie 😊.

Prerequisite

You have to install bochs and mtools in your linux system first. For Debian based distributions:

```
apt-get install bochs bochs-x mtools
```

is enough, for RPM based distributions, try:

```
yum install bochs mtools
```

In order to build linux kernel code, you must have gcc and make installed in your system. For Debian, you can run:

```
apt-get install build-essential bin86
```

Installation

Just untar the package:

```
tar xjf linux-0.11-lab_060618_005627.tar.bz2
```

then you will see new linux-0.11 directory under current working directory.

Working Flow

The working flow is really simple, build the kernel, install the kernel, and then run the kernel in bochs.

Assumes you are in linux-0.11 directory,

```
cd source/linux
make clean
make
cd ../../
```

will create a new file `Image' in source/linux directory, then

```
./mcopy_kernel source/linux/Image bootimage-new
```

will copy Image file into a bootable floppy disk with grub installed. you can find a new image file named /boot/bootimage-new in that disk. Note that all image will be copied to /boot directory if you use this command.

Then you can run

```
./edit_menu.lst
```

to edit the configuration of grub to add new item for you kernel in grub menu. The grammar it's really simple.

You can add following 4 lines at the end of menu.lst:

```
title          Linux 0.11 compiled by gcc 4.1
root           (fd0)
kernel        /boot/bootimage-new
boot
```

At last you can run

```
./linux-0.11
```

to start bochs virtual machine, If no problem occurs, you will see the grub menu. Choose the newly created item to test your modification.

If you don't want to build linux-0.11 code in your system, you can still build linux-0.11 inside bochs vm. If you choose to work this way, my flash is recommended.

<http://www.quinnli.com/upload/linux-0.11.swf>

Directories

source/

linux/

 this is my modified version

linux-0.11-040327-rh9.tar.gz

 this is original code provided by Dr. Zhao

linux-0.11-040327-rh9.diff.gz

this is original diff file to compile under RH9

linux-0.11-060617-gcc4-diff.gz

this is my modification, most of them are trivial ones just to please gcc, except for one. Please refer to: (in chinese)

<http://www.quinnli.com/blog/archives/50>

conf/

configuration files

images/

floppy disk and hard disk images to be used in bochs. The hard disk is provided by www.oldlinux.org with full linux develop envirement. One of the floppy disks is a grub disk, you can copy your new kernel Image into this floppy disk.

bak/

Your grub disk image is backedup here every time you use `./edit_menu.lst` or `./mcopy_kernel` incase you want to restore it some times.

Use other versions of gcc

Versions other than gcc-4.1 are also supported, gcc-4.0 should works without problem, and if you have an old enough distribution with gcc-2.95, congratulations to you, it is also supported.

But if you have gcc-3.x as your default gcc version, you don' t have to use my modification at all, just use the original one provided by Dr. Zhao. I also included his version in the package for convenience.

If you really want to use my modification to compile under gcc-3.x, that' s ok too, just overwrite ``mm/Makefile'` with ``mm/Makefile-gcc3.x'` . The original ``mm/Makefile'` have a backup ``mm/Makefile-gcc-4.1'` . Do REMEMDER this because you won' t face any error if you compile the code without overwrite ``mm/Makefile'` , but you won' t be able to boot your system with your newly compiled Image.

Known Problems

1. My kernel Image won' t mount floppy rootdisk.

Yes, if you use current version(2.2.6) of bochs and mount you new Image as floppy disk a directly, this problem may occur, and I don' t know why if you are sure you' v inserted the rootimage in bochs floppy B.

But if you use grub to load your kernel Image as shown in this file, you may not face this problem at all. I don' t know why but my guesss is that grub did some thing to configure the floppy drive which linux 0.11 did not do.

2. My kernel image will only mount floppy b.

Yes, it' s the default root device. You can use rdev (must be root user to execute rdev) to change the root device for a image.

```
rdev source/linux/Image /dev/hda1
```

And then use ./mcopy_kernel to overwrite the original ones.

(It' s amazing rdev can still work with linux 0.11. Really good backward compatibility!)

3. My kernel image is booted, but a lot of command won' t work, and the error is `not owner`

Yes. If you use hard disk root image, you have a simple solution to this problem. Backup /bin/sh first, and overwrite it with /bin/sh0. It should be ok after reboot.

If you want to understand why, please see [here](#)

Technorati Tags: [linux](#), [kernel](#)

[Add comment](#) Jun 18, '06, 03:44am quinn. liqin

尝试用高版本 gcc 编译 0.11 源码出问题

用的代码是 Oldlinux 论坛上[发布的](#)可以在 Redhat 9 下面编译通过的[源码](#)。用当前 Debian Sid 上的 gcc 4.1 编译，需要改动部分代码。编译好之后运行，bochs报错：prefetch: RIP > CS.limit。貌似[其他地方](#)也有碰到类似的问题，但是没有明确的解决方法。论坛上也有[一](#)则相关的[讨论](#)，不过没有明确讨论结果。在代码中插入一些 printf 之后发现是在 init/main.c 文件中 move_to_user_mode() 之后出错，不知道具体位置，因为不知道怎么调试：（。

没办法，只好想办法在 Debian Sid 上装了一个 gcc-2.95，经典编译器。重新编译后运行，这回 bochs 不报错了，Linux 本身报错：

```
out of memory
out of memory
BAD BAD - no father found
out of memory
```

这个也应该是在 init/main.c 文件中的 init 函数中出错。

郁闷了，不知道怎么调试bochs或者反汇编内核Image，搞不下去了，看看有没有办法搞出一个类似 RH9 的编译环境，按理说装了 gcc 2.95 已经可以了，莫非是 ld 的版本问题？下面试试在 Debian Woody上编译，没准成。BTW：貌似bochs在Windows下有图形调试界面，这年头，真是本末倒置了。

补充：尝试装了一个woody，默认编译器是gcc2.95，仍然和Sid上用2.95编译出现一样的错误，out of memory，最后发现原来用gcc 3.x（已经尝试3.0和3.3）编译就没有问题，看样子原来就是修改到gcc3.x下的。还是没搞清楚为什么用gcc2.95和gcc4.1编译会有问题。先研究一下牛人移植到gcc 3.x下的[diff文件](#)，看看能不能看出什么门道来。

补充2：问题基本已经解决，表层原因是 gcc2.95, gcc3.3, gcc4.1 在编译内核内存分配模块时 mm 过度优化的结果，主要是 mm/memory.c 文件的编译选项需要好好斟酌。经过测试 gcc2.95 和 gcc4.1 需要用下面的参数：

```
CFLAGS =-Wall -O -fstrength-reduce -fomit-frame-pointer -nostdinc -I../include
```

gcc-3.3 需要加上 `-finline-functions` 参数

```
CFLAGS =-Wall -O -fstrength-reduce -fomit-frame-pointer -finline-functions -nostdinc
-I../include
```

就可以编译出可用的内核。这个内核应该配合[原始的linux 0.11 rootdisk](#)使用，或者至少将根文件系统中的/bin/sh替换为0.11 rootdisk中的版本（0.11需要与不支持job control的bash配合使用，不然会出现各种各样的古怪问题，请参考论坛中的[原帖](#)）

[上传 linux-0.11-lab 到我的网站，大家一起学啊](#)

有两个版本，Windows的用户请[这里下载](#)，Linux的用户请[这里下载](#)。下载后解压先。还专门为Windows用户作了一个Flash教程，没办法，Windows下面懒人多。点[这里](#)看之。

Linux 用户请听我接着说。首先保证先装了 Bochs，和 mtools，如果是 Debian，可以

```
apt-get install bochs-* mtools
```

然后在**图形界面**下进入解压后的目录，启动 linux-0.11

```
cd linux-0.11-lab  
./linux-0.11
```

之后就应该出现 bochs 虚拟机的界面，显示 grub 的菜单。下面的内容可以参考[Flash教程](#)

学习 Linux 内核，是学最新的好？还是学最老的好？

想学 Linux 内核的想法在心中持续好多年，一直没有找到机会好好学。原因可能是多种多样的：首先是没有恒心，学习内核不是一朝一夕的事情，是一件系统工程，需要系统的掌握各种专业技能（参见[赵炯博士列的书单](#)），任何人都应该把学习 Linux 作为一个博士课题来开展，我往往是坚持了一两天就放了下来；其次是没有特定的目标，学习 Linux 内核这种事情，如果没有目标，往往就会被淹没在背景资料的海洋里无法自拔，看书的战线拉的过长，到时候全部溃不成军，没有多少收获。最后是学习 Linux 不能只停留在学上，必须理论和实践结合，实践有两个层面，首先是要用 Linux，在 Windows 平台学 Linux 永远是雾里看花，其次是要实验，如果代码只看不编译，不随便的改吧改吧，那还学个什么劲，不如看小说乐得轻松。

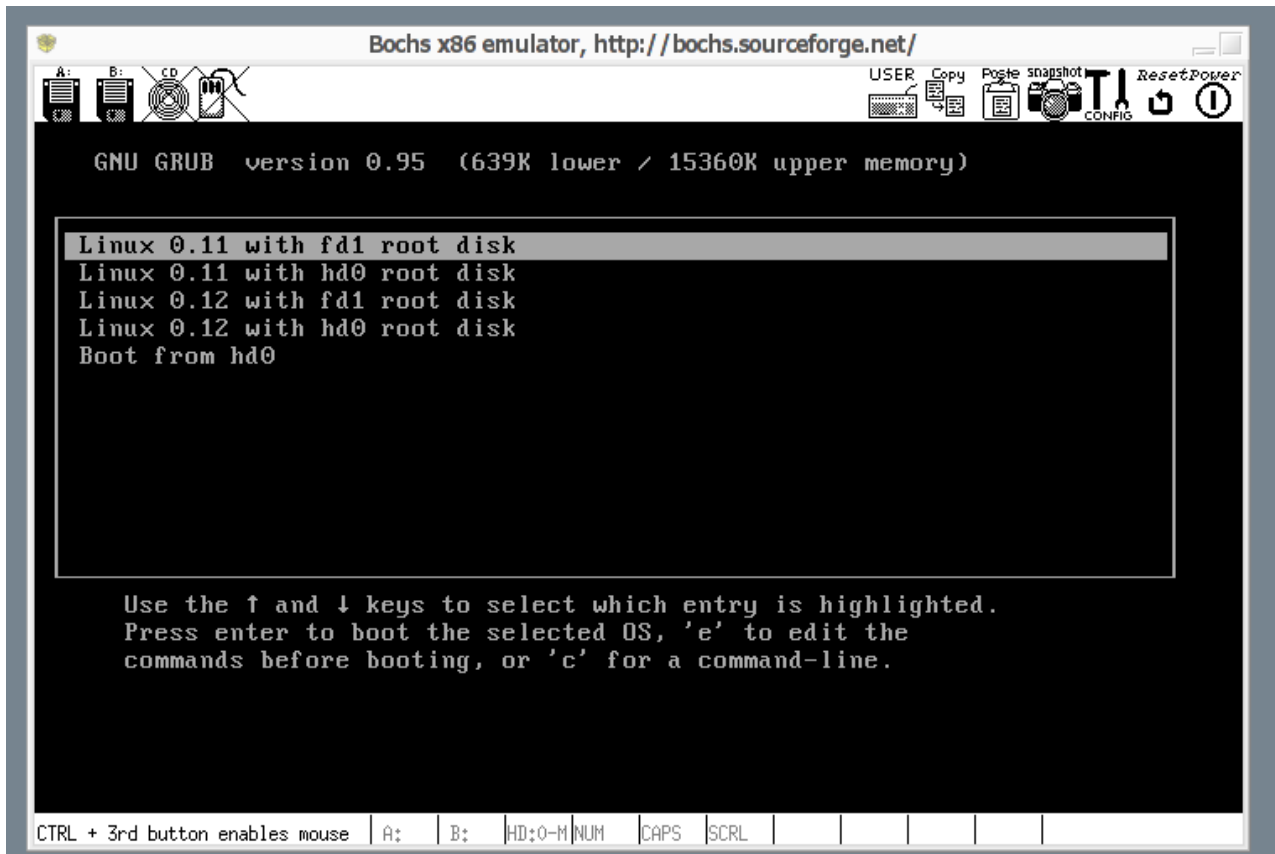
现在的 Linux 越来越大，也越来越难学了（Linux 2.6.16 内核总行数已经是触目惊心的 600 万行，除去其它平台代码和所有驱动程序代码的 i386 平台相关代码是 200 万行），一个人穷尽其一生，也不一定能全部学完，即使是 Linus 本人肯定也不是全懂现在这个庞然大物。于是乎，很多人就不约而同的有了一个想法，学习现在的 Linux 可能很难，学习最初的 Linux 总该很容易了吧，我自然也不例外。

于是我就想去找 [Linux 0.01](#)，也就是 Linus 本人最初在 Minix 邮件列表中发布的那个版本（实际上 Linus [第一个 Announce 的版本](#)是 0.02，而并没有在邮件列表中 Announce 0.01 这个版本，而只是私下发了通知给 Minix 讨论组中感兴趣的人，[Linus 本人的解释](#)是对版本 0.01 并不是非常自信，老大真是谦虚啊）。翻遍了整个 Internet 也只是在 LKML 里面找到[一个线索](#)是讨论如何恢复 0.01 的运行环境的，看了以后非常受打击，原来那时候 Linux 非常不完善，还需要那时候的 Minix 作为宿主编译开发环境，那时候的 gcc 版本是 1.4，而且上哪里去找这些个古董呢。

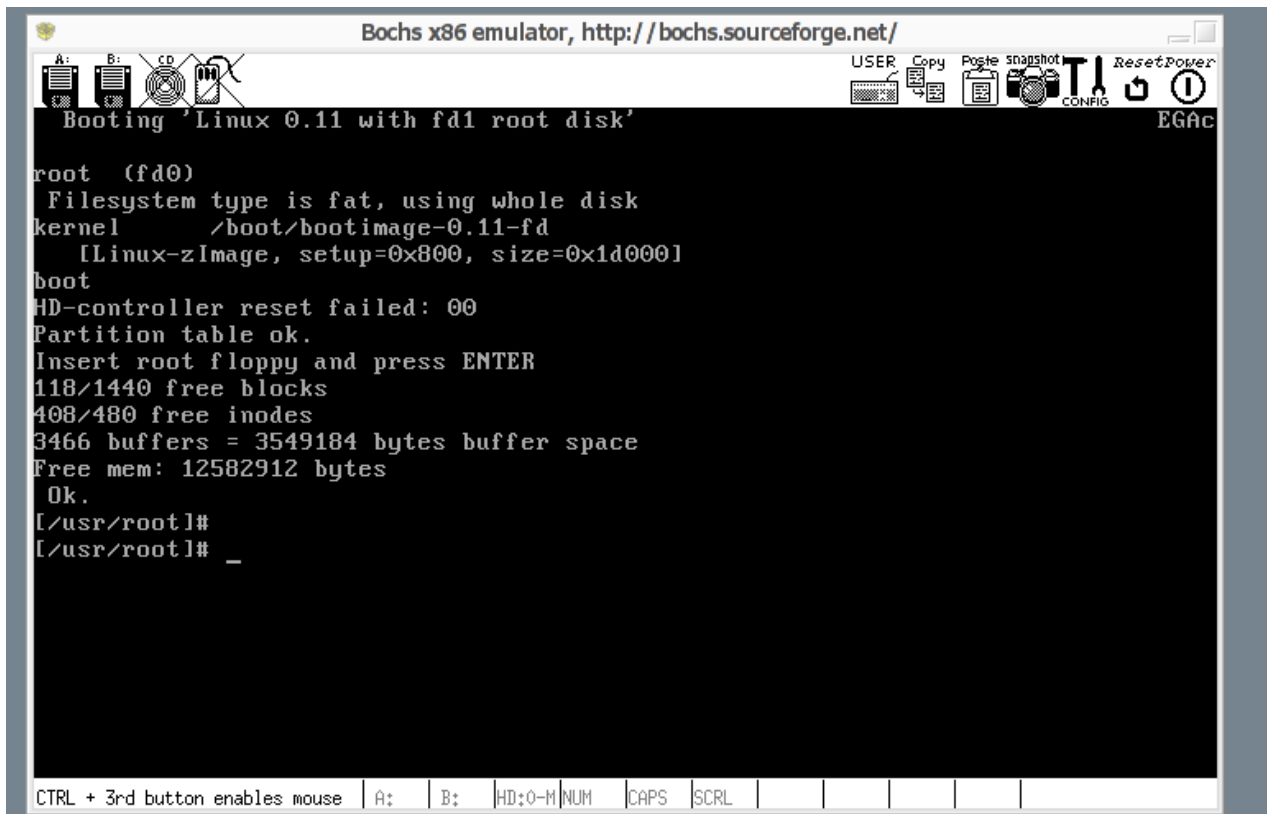
然后我就在 Google 上排列组合关键字做海量数据查询，最后终于被我发现了 [OldLinux](#) 这个网站，而且惊讶的发现（太后知后觉了）站长是中国人，同济大学的[赵炯博士](#)。OldLinux 这个网站应该算是颇有威望，Kernel.org 还对它做了[镜像](#)。我对赵炯博士写的好书[《Linux内核 0.11\(0.95\)完全注释》](#)（[下载](#)）真是相见恨晚，这本书顾名思义，自然是分析 0.11 版的Linux。之所以选择 0.11 版进行分析学习也是有深层次原因的，因为 0.11 版的Linux已经足够成熟，可以运行gcc编译自身了，但是又不至于过于复杂，导致代码量过大，[0.11 的代码只有区区 14000 行不到](#)（包括汇编）。BTW：从 1991 年 10 月 5 日发布 0.02 到 1991 年 12 月 19 日发布 0.11 仅用了两个半月，这种进步只能用神速来形容，能做出这种进步的只能是神仙了。另外一个值得一提的是，0.11 版本的Linux还不能算是开源软件，当时是 Linus [自己定的 License](#)，从[0.12 版本才开始](#)使用 GNU Copyleft。

OldLinux 上提供了非常丰富的Linux版本 1.0 之前的[资源](#)，其中包括完整的 0.11 和 0.12 编译环境，有兴趣的话可以去他的[论坛](#)上看使用说明。提供的编译环境运行在 Bochs 虚拟机里，在 Windows 下面也可以学习 Linux 源码，并且可以修改和重新编译。唯一的遗憾是主要目标用户是 Windows 用户，Linux 用户还需要修改一些配置文件。花了几天的时间，我也把 Linux 下面的 0.11 实验环境架好了，下面是截图：

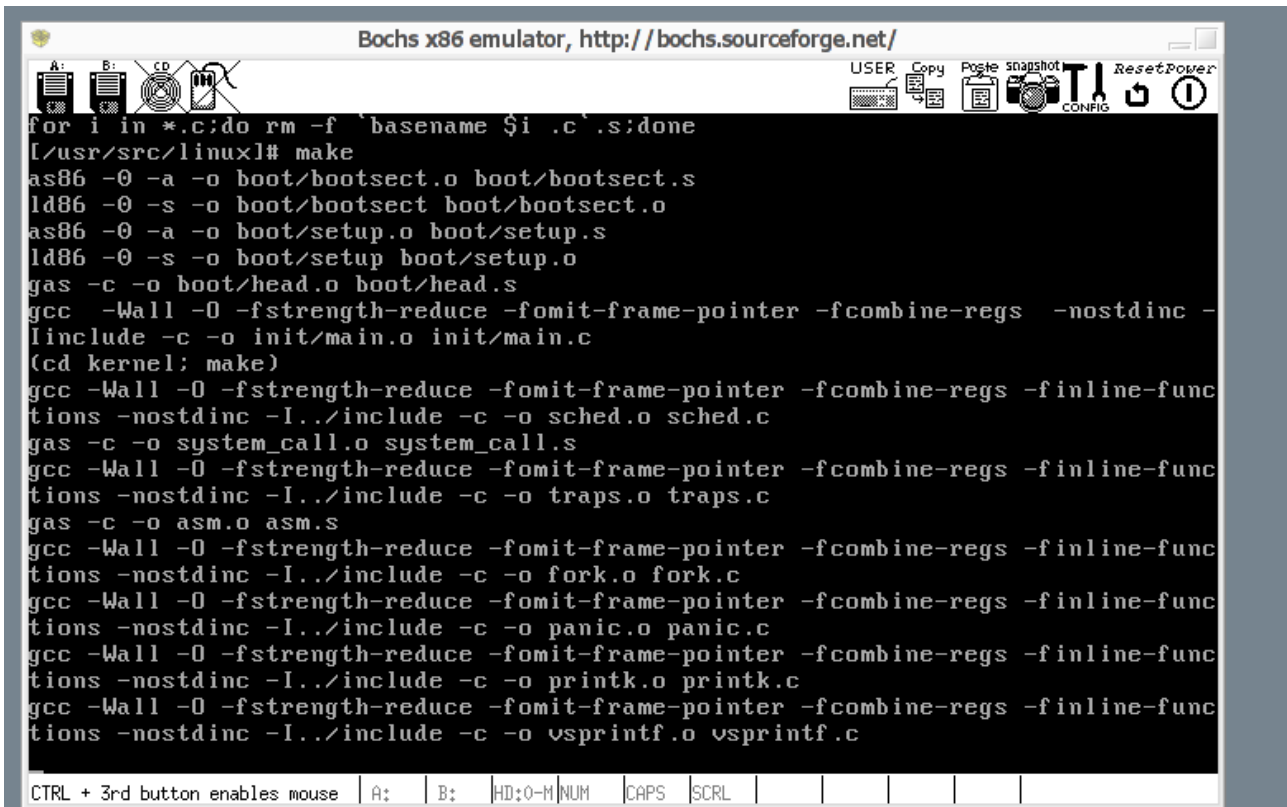
- 自己作了一个 Grub 引导盘



- 进入 Linux 0.11



- 在 Linux 0.11 下面编译 Linux 0.11 的源码



```
Bochs x86 emulator, http://bochs.sourceforge.net/
A: B: CD
USER Copy Paste Snapshot CONFIG Reset Power
for i in *.c;do rm -f `basename $i .c`.s;done
[/usr/src/linux]# make
as86 -O -a -o boot/bootsect.o boot/bootsect.s
ld86 -O -s -o boot/bootsect boot/bootsect.o
as86 -O -a -o boot/setup.o boot/setup.s
ld86 -O -s -o boot/setup boot/setup.o
gas -c -o boot/head.o boot/head.s
gcc -Wall -O -fstrength-reduce -fomit-frame-pointer -fcombine-regs -nostdinc -
linclude -c -o init/main.o init/main.c
(cd kernel; make)
gcc -Wall -O -fstrength-reduce -fomit-frame-pointer -fcombine-regs -finline-func
tions -nostdinc -I../include -c -o sched.o sched.c
gas -c -o system_call.o system_call.s
gcc -Wall -O -fstrength-reduce -fomit-frame-pointer -fcombine-regs -finline-func
tions -nostdinc -I../include -c -o traps.o traps.c
gas -c -o asm.o asm.s
gcc -Wall -O -fstrength-reduce -fomit-frame-pointer -fcombine-regs -finline-func
tions -nostdinc -I../include -c -o fork.o fork.c
gcc -Wall -O -fstrength-reduce -fomit-frame-pointer -fcombine-regs -finline-func
tions -nostdinc -I../include -c -o panic.o panic.c
gcc -Wall -O -fstrength-reduce -fomit-frame-pointer -fcombine-regs -finline-func
tions -nostdinc -I../include -c -o printk.o printk.c
gcc -Wall -O -fstrength-reduce -fomit-frame-pointer -fcombine-regs -finline-func
tions -nostdinc -I../include -c -o vsprintf.o vsprintf.c
CTRL + 3rd button enables mouse | A: | B: | HD:0-M | NUM | CAPS | SCRL | | | | | |
```