实验环境的搭建与使用

本操作系统实验的硬件环境是IA-32(x86)架构的PC机（就是你现在正在使用的计算机），主要软件环境是Bochs + gcc + 你最喜欢的编辑器/IDE + 你最喜欢的操作系统 + Linux 0.11源代码。实验的基本流程是根据实验要求编写应用程序、修改Linux 0.11的源代码，用gcc编译后，在Bochs的虚拟环境中运行、调试目标代码。

上述实验环境涉及到的软件都是免费且开源的，具有较强的可移植性，可以在多种计算机的多种操作系统上搭建。为方便实验者，我们在最常见的两个平台——Windows和Ubuntu（最流行的GNU/Linux发行版之一）——上制作了hit-oslab集成环境。它基本包含了实验所需的所有软件，安装过程非常简单。对于其他平台，可参考集成环境搭建实验环境，本书不做介绍。

主要平台和工具简介

x86模拟器Bochs

Bochs是一个免费且开放源代码的IA-32(x86)架构PC机模拟器。在它模拟出的环境中可以运行Linux、DOS和各种版本的Windows等多种操作系统。而Bochs本身具有很高的移植性，可以运行在多种软硬件平台之上，这也是我们选择它做为本书的指定模拟器的主要原因。如果您想拥抱自由的Linux，那么Bochs几乎是您的不二选择。如果您想继续把自己绑定在Windows平台上，那么除了Bochs，您还可以选用VMware或者Microsoft Virtual PC。它们是最著名虚拟机软件，而且都可以免费使用。因为Bochs的是模拟器，其原理决定了它的运行效率会低于虚拟机。但对于本书所设计的实验来说，效率上的差别很不明显。而且，Bochs有虚拟机无可比拟的调试操作系统的能力，所以我们更建议您选用Bochs。hit-oslab已经内置了bochs，本书后文假定的缺省环境也是Bochs。

关于Bochs的更详细的介绍请访问它的主页及Bochs使用手册。

GCC编译器

GCC是和Linux一起成长起来的编译器。Linux最初的版本就是由GCC编译的。现在GCC也是在自由软件领域应用最广泛的编译器。所以，我们也选择GCC做为本书实验的指定编译器。

DB调试器

GDB调试器是GCC编译器的兄弟。做为自由软件领域几乎是唯一的调试器，它秉承了*nix类操作系统的一贯风格，采用纯命令行操作，有点儿类似dos下的debug。关于它的使用方法请看*GDB使用手册*。

Ubuntu (GNU/Linux)

Ubuntu也许不是目前最好用的Linux桌面发行版，但它一定是最流行的。主要特点是易用，非常的易用。

现在，已经有越来越多的人开始用Ubuntu完全代替Windows，享受更加自由、安全、守法的感觉。Ubuntu的主页是http://www.ubuntu.com/，这里不仅可以免费下载到iso文件，甚至能免费申领Ubuntu的安装光盘。

我们强烈建议您在Ubuntu下做实验。因为有些实验内容涉及到在自己改进的Linux 0.11下，运行自己编的应用程序。被改进的功能都是高版本Linux内核已经具有的，在其上确认自己编写的应用程序无误后，再用之测试自己改进的Linux 0.11，可以更有信心些。

Microsoft Windows

人类历史上最重要的操作系统之一，用户最多的操作系统（没有之一），天下谁人不识君。

源代码阅读工具

实验过程中不可避免地要阅读Linux 0.11的源代码。源代码阅读工具能让这个过程的效率更高，烦恼更少。

http://www.oldlinux.org/lxr/http/source/ 是用lxr生成的一个Linux源代码阅读网站。只要用鼠标点击，就能轻松浏览Linux 0.11的源代码。

但在不能上网的时候，或者阅读的同时还要修改源代码，就要使用下面介绍的工具了。

在Linux下，ctags可以和VIM或Emacs配合，一边浏览代码，一边编辑。在Ubuntu下这样安装：

```
$ sudo apt-get install exuberant-ctags
```

使用起来也不麻烦，网上资料很多，man文档也很详细。

在Windows下，Source Insight是个不错的选择。

CodeView-oslab集成环境

0603102班的高汉东同学在使用hit-oslab后，对它深恶痛绝，他是这么说的：

“……在阅读linux-0.11源代码的的过程中，我们经常遇到一个问题：在一个文件中调用的函数并不在这个文件中定义，即所调用的函数是一个外部函数。为了找到这个函数的定义，我们要不就翻遍文件夹内所有的文件，要不就在“内核注释”中搜索。这两种方法都可以解决，但是很麻烦。……”

于是，他伙同几名同学，开发了CodeView，并和oslab深度集成。该软件为开源软件，支持Windows和Linux，不仅欢迎下载，而且欢迎有识之士提交建议、补丁，甚至加盟。

实验环境的工作模式

hit-oslab实验环境简称oslab，可在“资料和文件”下载。oslab工作在一个宿主操作系统之上，分为Linux版和Win32版。其中Linux版主要针对Ubuntu的32位和64位版设计，但稍加修改甚至完全不修改，也可以在其它Linux发行版使用。这是我们主要推荐的版本。Win32版主要针对Windows XP设计，不能很好地支持Windows Vista。

在宿主操作系统之上完成对Linux 0.11的开发、修改和编译之后，在linux-0.11目录下会生产一个名为Image的文件，它就是编译之后的目标文件。该文件内已经包含引导和所有内核的二进制代码。如果拿来一张软盘，从它的0扇区开始，逐字节写入Image文件的内容，就可以用这张软盘启动一台真正的计算机，并进入Linux 0.11内核。oslab采用bochs模拟器加载这个文件，模拟执行Linux 0.11，这样省却了重新启动计算机的麻烦。

bochs目录下是与bochs相关的执行文件、数据文件和配置文件。run是运行bochs的脚本命令。运行后bochs会自动在它的虚拟软驱A和虚拟硬盘上各挂载一个镜像文件，软驱上挂载是linux-0.11/Image，硬盘上挂载的是hdc-0.11.img。因为bochs配置文件中的设置是从软驱A启动，所以Linux 0.11会被自动加载。而Linux 0.11会驱动硬盘，并mount硬盘上的文件系统，也就是将hdc-0.11.img内镜像的文件系统挂载到0.11系统内的根目录——“/”。在0.11下访问文件系统，访问的就是hdc-0.11.img文件内虚拟的文件系统。

hdc-0.11.img文件的格式是Minix文件系统的镜像。Linux所有版本都支持这种格式的文件系统，所以可以直接在宿主Linux上通过mount命令访问此文件内的文件，达到宿主系统和bochs内运行的Linux 0.11之间交换文件的效果。Windows下目前没有（或者是还没发现）直接访问Minix文件系统的办法，所以要借助于fdb.img，这是一个1.44M软盘的镜像文件，内部是FAT12文件系统。将它挂载到bochs的软驱B，就可以在0.11中访问它。而通过filedisk或者WinImage，可以在Windows下访问它内部的文件。

hdc-0.11.img内包含有：

- Bash shell

- 一些基本的Linux命令、工具，比如cp、rm、mv、tar。
- vi编辑器
- gcc 1.4编译器，可用来编译标准C程序
- as86和ld86
- Linux 0.11的源代码，可在0.11下编译，然后覆盖现有的二进制内核

Linux上的实验环境

感谢兰州大学的falcon，他写了一篇详细的文章介绍如何在Linux下搭建Linux 0.11的实验环境，并且给出了相应的补丁。本环境大量参考和继承了他所做的工作。

环境搭建

Ubuntu下环境搭建

首先，到“资料和文件”中下载“hit-oslab-linux-XXXXXXXX.tar.gz”。此文件包中包括下列内容：

- 可在Ubuntu下编译的Linux 0.11内核源代码。
- 已打开gdb-stub功能的Bochs及其支撑文件、配置文件。Ubuntu自带的Bochs没有gdb-stub功能，不能用GDB进行C语言级的调试。
- 可忽略Signal 0的GDB调试器。GDB是为调试应用程序而设计，而应用程序不需要处理signal 0，所以GDB捕获到signal 0后会强制暂停程序。Bochs（也许是Linux 0.11内核）会产生大量的signal 0，影响调试。我们给GDB打了一个补丁，使其可以忽略signal 0。
- 磁盘镜像文件hdc-0.11-new.img。它是Linux 0.11的根文件系统，内含gcc、vi等开发工具和bash等常用工具。
- 方便运行、调试和文件交换的一系列脚本。

建议将oslab.tar.gz保存到你的home目录(/home/xxxx (xxxx是你的用户名))下。然后打开终端窗口（左上角的菜单，Applications->Accessories->Terminal），当前目录就是你的home目录。这个窗口是将来要使用的主要窗口。在提示符下执行如下命令（不包括那个“$”）解压缩下载的文件：

```
$ tar zxf oslab.tar.gz
```

用ls命令列目录如果能看到“oslab”目录，就说明解压缩成功。这个目录下已经包括Linux-0.11源代码、Bochs、GDB和一些数据文件及脚本。但我们还需要安装编译器和编辑器等开发环境。

首先安装gcc-3.4。因为Linux-0.11不能在gcc 4.x版本编译，所以要装老一点儿的编译器。在Ubuntu 9.04(jaunty)及之前，用下面命令安装：

```
$ sudo apt-get install gcc-3.4
```

在Ubuntu 9.10(karmic)及之后，要先从“资料和文件”中下载gcc-3.4-ubuntu.tar.gz到/tmp目录，用下列命令安装：

```
$ cd /tmp
$ tar zxvf gcc-3.4-ubuntu.tar.gz
$ cd gcc-3.4
$ sudo ./inst.sh xxxx #xxxx换为i386或amd64
```

然后执行下列命令确保其它必备工具已安装：

```
$ sudo apt-get install build-essential bin86 manpages-dev
```

如果使用的是64位的系统，还要：

```
$ sudo apt-get install libc6-dev-i386 ia32-libs ia32-libs-gtk
```

还要安装你喜欢（或将来会喜欢）的编辑器/IDE。我们推荐vim：

```
$ sudo apt-get install vim cscope exuberant-ctags
```

也推荐emacs：

```
$ sudo apt-get install emacs
```

“vimtutor”是一个非常简单直接的vim入门培训命令，安装了vim后，它就已经存在，可以试试。

Linux上肯定也有类似Windows的编辑器，但并不推荐它们，因为真正的Linuxer只喜欢vi或者emacs。

选择了一种编辑器，就选择了一种文化和生活习惯，所以请慎重选择。

至此，环境搭建完毕。

Fedora下环境搭建

本节由计算机学院09级杨靖同学撰写。

首先，请参考Ubuntu实验环境的搭建，下载并解压hit-oslab包。

之后是安装必要的工具：gcc的3.4版本，因为linux-0.11不能在gcc 4.x下编译。 在Fedora下，用下列命令安装：

```
$sudo yum install compat-gcc-34 glibc-devel libgcc
```

如果没有sudo权限，则运行

```
$su -c "yum install compat-gcc-34 glibc-devel libgcc"
```

以下命令类似，如果不能sudo则su。su命令需要root密码，（应该）是在安装Fedora的时候指定的。

如果是64位系统，那么应该指定安装32位的库。 在64位Fedora下运行如下命令来安装gcc 3.4

```
$sudo yum install compat-gcc-34 glibc-devel.i686 libgcc.i686
```

安装完成之后再以普通用户身份运行

```
$ln -s /usr/bin/gcc34 ~/bin/gcc-3.4
```

然后是其他必备工具

```
$sudo yum install dev86
```

还有一些必要的库，

如果是32位系统

```
$sudo yum install libSM libX11 libXpm libstdc++ ncurses-libs expat
```

如果是64位的系统

```
$sudo yum install libSM.i686 libX11.i686 libXpm.i686 libstdc++.i686 ncurses-libs.i686 expat.i686
```

即使安装了这些软件包，实验时还是有可能出现找不到某些lib的情况。

遇到这种情况请先读错误输出，从中找出缺少的库（一般都是libxxx.so.xxx），然后尝试

1. 输入yum search 缺少库的名字(例如libXpm则输入Xpm)
2. Google “fedora libxxx”
3. 到cms的论坛上发帖求助

这三种方法来找到所需的软件包并安装上。

另外，在Fedora与linux-0.11交换文件时，可能需要暂时关闭SElinux，否则SElinux会阻止用户的某些磁盘写入操作。

使用方法

准备活动

```
$ cd ~/oslab
```

把当前目录切换到oslab下，用pwd命令确认，用“ls -l”列目录内容。本实验的所有内容都在本目录或其下级目录内完成。

编译内核

“编译内核”比“编写内核”要简单得多。首先要进入linux-0.11目录，然后执行：

```
$ make all
```

因为“all”是最常用的参数，所以可以省略，只用“make”，效果一样。

在多处理器的系统上，可以用-j参数进行并行编译，加快速度。例如双CPU的系统可以：

```
$ make -j 2
```

make命令会显示很多很多很多的信息，你可以尽量去看懂，也可以装作没看见。只要最后几行中没有“error”就说明编译成功。最后生成的目标文件是一个软盘镜像文件——linux-0.11/Image。如果将此镜像文件写到一张1.44MB的软盘上，就可以启动一台真正的计算机。

linux-0.11目录下是全部的源代码，很多实验内容都是要靠修改这些代码来完成。修改后需要重新编译内核，还是执行命令：

```
$ make all
```

make命令会自动跳过未被修改的文件，链接时直接使用上次编译生成的目标文件，从而节约编译时间。但如果重新编译后，你的修改貌似没有生效，可以试试先“make clean”，再“make all”。“make clean”是删除上一次编译生成的所有中间文件和目标文件，确保是在全新的状态下编译整个工程。

运行和调试

在Bochs中运行最新编译好的内核很简单，在oslab目录下执行：

```
$ ./run
```

如果出现Bochs的窗口，里面显示linux的引导过程，最后停止在“[/usr/root/]#”，表示运行成功。

内核调试分为两种模式：汇编级调试和C语言级调试。

汇编级调试需要执行命令：

```
$ ./dbg-asm
```

可以用命令help来查看调试系统用的基本命令。更详细的信息请查阅Bochs使用手册。

C语言级调试稍微复杂一些。首先执行如下命令：

```
$ ./dbg-c
```

然后再打开一个终端窗口，进入oslab目录后，执行：

```
$ ./rungdb
```

新终端窗口中运行的是GDB调试器。关于gdb调试器请查阅*GDB使用手册*。

Ubuntu和Linux 0.11之间的文件交换

oslab下的hdc-0.11-new.img是0.11内核启动后的根文件系统镜像文件，相当于在bochs虚拟机里装载的硬盘。在Ubuntu上访问其内容的方法是：

```
$ sudo ./mount-hdc
```

（在格物楼机房，直接用“mount hdc”代替上面命令，不需sudo。但要求oslab必须在/home/public_user/oslab或/public/oslab）

之后，hdc目录下就是和0.11内核一模一样的文件系统了，可以读写任何文件（可能有些文件要用sudo才能访问）。读写完毕，不要忘了卸载这个文件系统：

```
$ sudo umount hdc
```

注意1：不要在0.11内核运行的时候mount镜像文件，否则可能会损坏文件系统。同理，也不要在已经mount的时候运行0.11内核。

注意2：在关闭Bochs之前，需要先在0.11的命令行运行“sync”，确保所有缓存数据都存盘后，再关闭Bochs。

Windows上的实验环境

（从2008年9月起，Windows版的hit-oslab停止维护和支持。使用中遇到的问题请自行解决。我们不对该版本造成的任何后果负责）

感谢flyfish，他写了一篇详细的文章介绍如何在Windows下搭建Linux 0.11的实验环境，并且给出了相应的代码。本环境大量参考和继承了他所做的工作。

环境搭建

首先，到“资料和文件”中下载最新版的hit-oslab-win32-XXXXXXXX.tar.gz。这个文件包中包括下列内容：

- 可在Windows下编译的Linux 0.11内核源代码。
- BIN86、GDB和GCC(MinGW)。
- 已打开gdb-stub功能，且不会发出signal 0的Bochs及其支撑文件、配置文件。Bochs的二进制发行版没有gdb-stub功能，不能用GDB进行C语言级的调试。而GDB和Bochs协同进行远程调试的时候，Bochs会发出大量的signal 0，影响调试。根据roy_hu在[http://www.oldlinux.org/oldlinux/viewthread.php?tid=10761]所述，我们给Bochs打了一个补丁，解决了此问题。
- 硬盘镜像文件hdc-0.11-new.img。它是Linux 0.11的根文件系统，内含gcc、vi等开发工具和bash等常用工具。

- 软盘镜像文件fdb.img。用来在Windows和Linux 0.11之间交换文件。
- filedisk，在Windows上将fdb.img虚拟成为一个可直接访问的磁盘。
- 方便运行、调试的一系列批处理文件。

将oslab解压后，进入oslab文件夹，可以看到文件夹下已经包括Linux-0.11源代码、Bochs、MinGW和一些数据文件及批处理文件。

为了能在Windows NT/2000/2003/XP和虚拟机中运行的Linux 0.11之间交换文件，需要安装filedisk。首先先进入filedisk文件夹，然后双击“setup”安装filedisk，最后根据提示重新启动系统。

Windows Vista以上用户不需做上一步。因为filedisk不支持Vista。Vista用户可以自行下载、安装商业软件WinImage([http://www.winimage.com/])

重启完毕，只需再安装一个你喜欢的编辑器或IDE，例如：UltraEdit、EditPlus等。请自行选择、下载、安装。

至此，环境搭建完毕。

使用方法

编译内核

“编译内核”比“编写内核”要简单得多，只需双击“make all”。它运行的时候会显示很多很多很多的信息，你可以尽量去看懂，也可以装作没看见。只要最后几行中没有“error”就说明编译成功。最后生成的目标文件是一个软盘镜像文件——linux-0.11\Image。如果将此镜像文件写到一张1.44MB的软盘上，它就可以用来启动一台真正的计算机。

linux-0.11目录下是全部的源代码，很多实验内容都是要靠修改这些代码来完成。修改后需要重新编译内核，这时最好先双击“clean”，它会删除上一次编译生成的所有中间文件和目标文件，然后再“make all”。

每次编译之前都做一次clean可以避免一些不必要的麻烦，但会增大编译时间。怎么做合适，就要靠经验决定了。

运行和调试

在Bochs中运行最新编译好的内核很简单，在oslab文件夹下双击“run”

如果出现Bochs的窗口，里面显示linux的引导过程，最后停止在“[/usr/root/]#”，表示运行成功。

内核调试分为两种模式：汇编级调试和C语言级调试。

汇编级调试需要运行“dbg-asm”。在Console窗口中可以用命令help来查看调试系统用的基本命令。更详细的信息请查阅Bochs使用手册。

C语言级调试稍微复杂一点点。

首先运行“dbg-c”。它会打开网络端口1234，所以如果安装了防火墙一类的软件，会弹出提示，一定要允许这个端口被访问。

然后运行“rungdb”，打开的命令行窗口中运行的就是GDB调试器。关于GDB调试器请查阅*GDB使用手册*。

Windows和Linux 0.11之间的文件交换

oslab下的fdb.img是一个1.44MB软盘镜像文件，Bochs启动时会自动挂载到b:。

在Windows NT/2000/2003/XP上我们通过filedisk的虚拟磁盘访问这个文件的内容，它是一款自由软件，免费且开放源代码。不过，它不支持Windows Vista，所以在Windows Vista下只能使用WinImage，它可以直接打开img文

件，就像打开一个压缩包一样。但它是商业软件，只有30天的免费试用期。出于版权的考虑，我们没有在实验环境中包含它。

如果使用filedisk，只需双击“mount”，系统会出现一个磁盘“X:”（就像插入一个U盘一样），这个磁盘就是fdb.img的虚拟磁盘，可以在其上读写任何文件。读写完毕，不要忘了双击“umount”，卸载这个文件系统。

在Linux 0.11环境里，输入命令：

```
# mcopy hello.c b:
```

可以将文件hello.c写入文件fdb.img中。而命令：

```
# mcopy b:hello.c .
```

则是从fdb.img拷贝到当前目录。其中mcopy来自Linux/UNIX读写MSDOS文件的一个工具mtools。

如果使用上述命令出现“Reset-floppy called”的错误，那么首先执行一次“mdir a:”，再执行一次“mdir b:”，然后就可以mcopy了。

要特别注意的是，当Bochs运行的时候不能mount，而未umount的时候，Bochs不能访问这个镜像文件。

另外，在关闭Bochs之前，需要先在0.11的命令行运行“sync”，确保所有缓存数据都存盘后，再关闭Bochs。

FAQ

1. **Q: 为何有时Bochs的光标闪动，却不响应我的输入？**
   A: 按一下Alt，然后再试试。
   如果你习惯用Alt+Tab切换窗口，就肯定会遇到这个问题。原因是在Bochs窗口按下Alt，Bochs会接收到Alt按下的事件，然后将此事件传给Linux 0.11。待再按下Tab时，主机操作系统经判断认定这是一个切换窗口的快捷键，于是直接切换窗口，Tab和Alt抬起的事件都不会再发给Bochs。等切换会Bochs，Linux 0.11此时还处于认为Alt已按下的状态，再按任何按键都被解释成是和Alt组合的按键，所以就“好像”不响应了（按数字键可以看到它的响应）。
2. **Q: 怎样加快make clean、make all的速度？**
   A: 如果只修改了kernel目录下的文件，删除kernel目录下的kernel.o，然后直接make就行了。其它目录方法类似。
3. **Q: Bochs屏幕乱了怎么办？**
   A: 这是Linux的终端控制和Bochs虚拟的终端之间配合不好导致的，一般在大量输出信息后，会出现混乱，甚至很像死机。此时按ctrl+l可以缓解一下。最好是用输出重定向功能将输出都重定向到一个文件，然后用vi看。