

Broncho 开发包文档

创建日期: **2008 年 3 月 26 日**

版本: **0.0.7**

更新日期: **2008 年 4 月 17 日**

目录

Broncho 开发包文档.....	1
目录.....	2
1. 修订历史.....	3
2. 概述.....	4
3. 开发包介绍.....	4
4. 开发应用程序.....	4
4.1 开发环境需求.....	4
4.2 开发环境搭建.....	6
4.2.1 虚拟机的安装 — Vmware.....	6
4.2.2 操作系统的安装 — Fedora core 7.....	9
4.3 编写程序.....	17
5. 开发复杂的程序.....	22
5.1 建立一个软件工程.....	22
5.2 将工程并入到 Broncho 工程.....	24
5.3 Broncho 工程的使用.....	24
6.Linux 系统使用基础.....	25
6.1 基本命令.....	25
6.2 GNU 编译工具.....	25
7.Broncho 工程详解.....	26
8. 附录.....	26

1.修订历史


版本号	修订内容	修订人	修订日期
0.0.1	新编制	李运斌	2008-03-26
0.0.2	充实内容	李运斌	2008-03-27
0.0.3	充实内容	李运斌	2008-03-28
0.0.4	修正内容	李运斌	2008-03-28
0.0.5	修正错误, 增加参考资料	李运斌	2008-03-31
0.0.6	修正错误, 字体	李运斌	2008-04-01
0.0.7	修正错误, 谢谢 i90091e@gmail.com	李运斌	2008-04-17


2.概述

Broncho 是一个基于 **linux** 核心，整合开源界优秀软件技术成果，拥有自创技术和架构方案的嵌入式平台。

本文档系统的介绍了开发 **broncho** 应用程序的一般过程，包括了环境的搭建、程序的编译和运行调试等等。通过阅读文档，让你可以快速掌握 **broncho** 应用程序的开发，编写自己的 **broncho** 应用程序。

在您阅读之前，我们先说明一下文档中的一些约定，让你能更好的理解文档。

 这个符号后的内容都是有用的提示。

 **# ls -al** #号后的内容都是运行在终端里的命令

 新出现的命令或用法，要留意了。

3.开发包介绍

Broncho SDK 镜像包含了开发 **broncho** 应用程序所需的操作系统、编译工具、链接库、头文件、文档和实用工具。**Broncho SDK** 的主要目录结构如下：

```
/broncho-sdk  
/broncho-sdk/tools  
/broncho-sdk/doc  
/broncho-sdk/example  
/broncho-sdk/work/pcemu  
/broncho-sdk/work/build/broncho
```

这些目录信息很重要，在接下来的学习中，你会用到的。记不住这些内容也没关系，可以随时回来阅读。在第四章节，我们开始向开发 **Broncho** 平台应用程序迈进。

4.开发应用程序

下面我们通过搭建 **Broncho** 平台开发环境和编写、运行一个简单的 **Broncho** 应用程序，来打开我们的 **Broncho** 开发之门。

4.1 开发环境需求

每个嵌入式平台对开发的环境都有一定要求，因为不同版本的软件会导致一些细微、烦人的问题。我们强烈推荐你使用我们推荐的开发环境，以减少开发过程中不必要的麻烦，待

完全熟悉 **Broncho** 的开发环境以后再根据自己的需求更变开发环境。本文档讲解的实例都是基于 **Broncho** 推荐的开发环境。

Broncho 开发环境基本需求:

最低配置:

CPU: Pentium 1.6GHz

MEM: 512MB

DISK: 30GB

推荐配置:

CPU: Pentium 2.0GHz

MEM: 1024MB

DISK: 160GB

因为开发时要运行的虚拟机是相当耗系统资源的，我们推荐你尽量使用较高配置的电脑，这样可以减少你开发时的痛苦缓慢和无聊等待。

Broncho 开发环境所需软件:

◇ **VMware-workstation-6.0.3-80004.i386.tar.gz**

◇ **Fedora core 7 的 vmware 镜像: vmware-fc7.tar.gz**

 这些你都可以从我们网站取得

<http://www.broncho.cn/download/sdk>

4.2 开发环境搭建

4.2.1 虚拟机的安装 — Vmware

这个虚拟机将会做为 **broncho** 开发环境的主要平台，程序的编写和编译、运行都将会在虚拟机里面进行。我们对虚拟机的操作，都通过 **ssh** 登录过去进行。根据我们以往的经验整个开发过程在虚拟机里完成的好处就是：

- 1) 防止开发过程的错误导致系统的崩溃，这个崩溃可能会给你带来很多“惊喜”，例如文件丢失、数据被破坏、系统死锁等等，这些在虚拟机里面我们都容易修复。
- 2) **Broncho** 应用程序运行在 **Directfb** 下，**Directfb** 会独占设备，这导致我们不能在 **Directfb** 和 **X-Window** 之间自由切换。

Vmware 的安装过程：

以 **VMware-workstation-6.0.3-80004.i386.tar.gz** 为例

- 1、开启一个 **shell** 终端窗口，进入 **vmware** 安装包所在的目录



```
lyb@lyb-desktop: /media/sda7/software
lyb@lyb-desktop: /media/sda7/software$ ls -l VMware-workstation-6.0.3-80004.i386.tar.gz
-rw----- 1 lyb lyb 222622982 2008-03-27 09:43 VMware-workstation-6.0.3-80004.i386.tar.gz
lyb@lyb-desktop: /media/sda7/software$
```

- 2、解压 **vmware** 安装包

🌟 输入命令：**# tar zxvf VMware-workstation-6.0.3-80004.i386.tar.gz**

```
lyb@lyb-desktop: /media/sda7/software
lyb@lyb-desktop:/media/sda7/software$ ls -l VMware-workstation-6.0.3-80004.i386.tar.gz
-rw----- 1 lyb lyb 222622982 2008-03-27 09:43 VMware-workstation-6.0.3-80004.i386.tar.gz
lyb@lyb-desktop:/media/sda7/software$ tar xzvf VMware-workstation-6.0.3-80004.i386.tar.gz
```

3、解压后会在 **vmware** 安装包所在的目录创建一个 **vmware-distrib** 目录，进入这个目录

🌟 输入命令: **# cd vmware-distrib**

```
lyb@lyb-desktop: /media/sda7/software/vmware-distrib
vmware-distrib/man/man1/
vmware-distrib/man/man1/vmware.1.gz
vmware-distrib/installer/
vmware-distrib/installer/services.sh
vmware-distrib/vmware-install.pl
vmware-distrib/FILES
lyb@lyb-desktop:/media/sda7/software$ ls -l vmware-distrib/
总用量 500
drwxr-xr-x 2 lyb lyb 4096 2008-03-04 11:03 bin
drwxr-xr-x 4 lyb lyb 4096 2008-03-04 11:03 doc
drwxr-xr-x 2 lyb lyb 4096 2008-03-04 11:03 etc
-r--r--r-- 1 lyb lyb 464292 2008-03-04 11:03 FILES
drwxr-xr-x 2 lyb lyb 4096 2008-03-04 11:03 installer
drwxr-xr-x 22 lyb lyb 4096 2008-03-04 11:03 lib
drwxr-xr-x 3 lyb lyb 4096 2008-03-04 11:03 man
drwxr-xr-x 2 lyb lyb 4096 2008-03-04 11:03 sbin
drwxr-xr-x 3 lyb lyb 4096 2008-03-04 11:03 system_etc
drwxr-xr-x 3 lyb lyb 4096 2008-03-04 11:03 usr
lrwxrwxrwx 1 lyb lyb 23 2008-03-27 09:54 vmware-install.pl -> bin/vmware-uninstall.pl
drwxr-xr-x 3 lyb lyb 4096 2008-03-04 11:03 vmware-vix
lyb@lyb-desktop:/media/sda7/software$ cd vmware-distrib/
lyb@lyb-desktop:/media/sda7/software/vmware-distrib$
```

4、运行 **vmware** 的安装脚本 **vmware-install.pl**

🌟 输入命令: **# ./vmware-install.pl**

```
ljb@ljb-desktop: /media/sda7/software/vmware-distrib
vmware-distrib/man/man1/
vmware-distrib/man/man1/vmware.1.gz
vmware-distrib/installer/
vmware-distrib/installer/services.sh
vmware-distrib/vmware-install.pl
vmware-distrib/FILES
ljb@ljb-desktop:/media/sda7/software$ ls -l vmware-distrib/
总用量 500
drwxr-xr-x  2 ljb ljb  4096 2008-03-04 11:03 bin
drwxr-xr-x  4 ljb ljb  4096 2008-03-04 11:03 doc
drwxr-xr-x  2 ljb ljb  4096 2008-03-04 11:03 etc
-r--r--r--  1 ljb ljb 464292 2008-03-04 11:03 FILES
drwxr-xr-x  2 ljb ljb  4096 2008-03-04 11:03 installer
drwxr-xr-x 22 ljb ljb  4096 2008-03-04 11:03 lib
drwxr-xr-x  3 ljb ljb  4096 2008-03-04 11:03 man
drwxr-xr-x  2 ljb ljb  4096 2008-03-04 11:03 sbin
drwxr-xr-x  3 ljb ljb  4096 2008-03-04 11:03 system_etc
drwxr-xr-x  3 ljb ljb  4096 2008-03-04 11:03 usr
lrwxrwxrwx  1 ljb ljb    23 2008-03-27 09:54 vmware-install.pl -> bin/vmware-uninstall.pl
drwxr-xr-x  3 ljb ljb  4096 2008-03-04 11:03 vmware-vix
ljb@ljb-desktop:/media/sda7/software$ cd vmware-distrib/
ljb@ljb-desktop:/media/sda7/software/vmware-distrib$ sudo ./vmware-install.pl
```

- 运行 **vmware-install.pl** 需要 **root** 权限
- 运行 **vmware-install.pl** 后会需要确认一系列的安装问题，直接“回车”使用默认值即可

5、安装完后，运行 **vmware** 测试是否安装成功

输入命令：**# vmware**

```
ljb@ljb-desktop: /media/sda7/software/vmware-distrib
Virtual machine monitor done
Blocking file system: done
Virtual ethernet done
Bridged networking on /dev/vmnet0 done
Host network detection done
Host-only networking on /dev/vmnet1 (background) done
DHCP server on /dev/vmnet1 done
Host-only networking on /dev/vmnet8 (background) done
DHCP server on /dev/vmnet8 done
NAT service on /dev/vmnet8 done

The configuration of VMware Workstation 6.0.3 build-80004 for Linux for this
running kernel completed successfully.

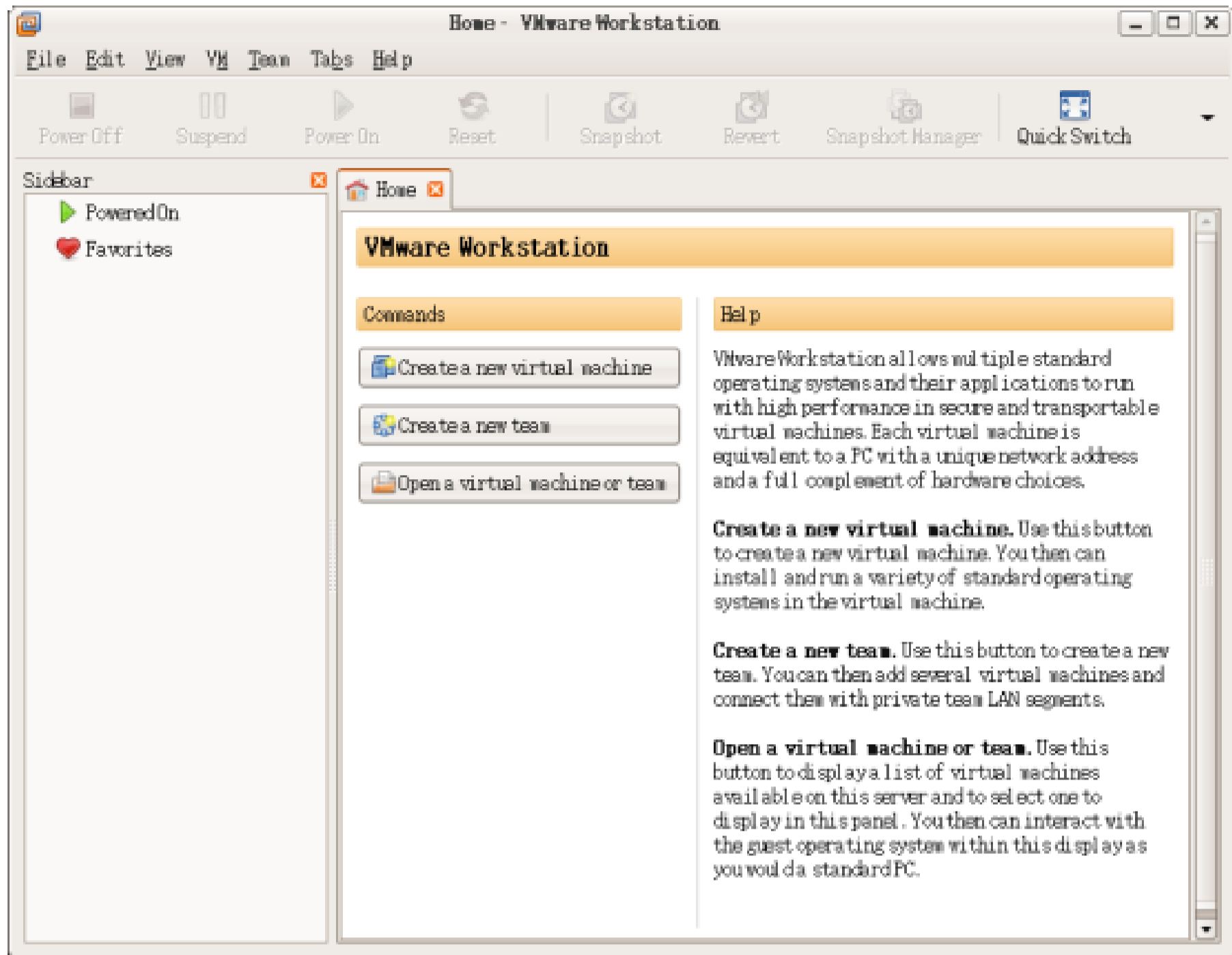
You can now run VMware Workstation by invoking the following command:
"/usr/bin/vmware".

Enjoy,

--the VMware team

ljb@ljb-desktop:/media/sda7/software/vmware-distrib$ vmware
```

6、出现了 **vmware** 的界面，恭喜您！**vmware** 安装成功。



4.2.2 操作系统的安装 — Fedora core 7

Broncho 平台开发使用 **Fedora core 7** 操作系统，使用的工具都是其上自带的工具。为了确保开发环境的一致，我们网站上已经提供了 **Fedora core 7** 的 **vmware** 镜像，直接在 **vmware** 里打开这个镜像使用即可。

- 1、开启一个 **shell** 终端窗口，进入 **vmware** 镜像所在的目录



2、解压我们提供的 vmware 镜像

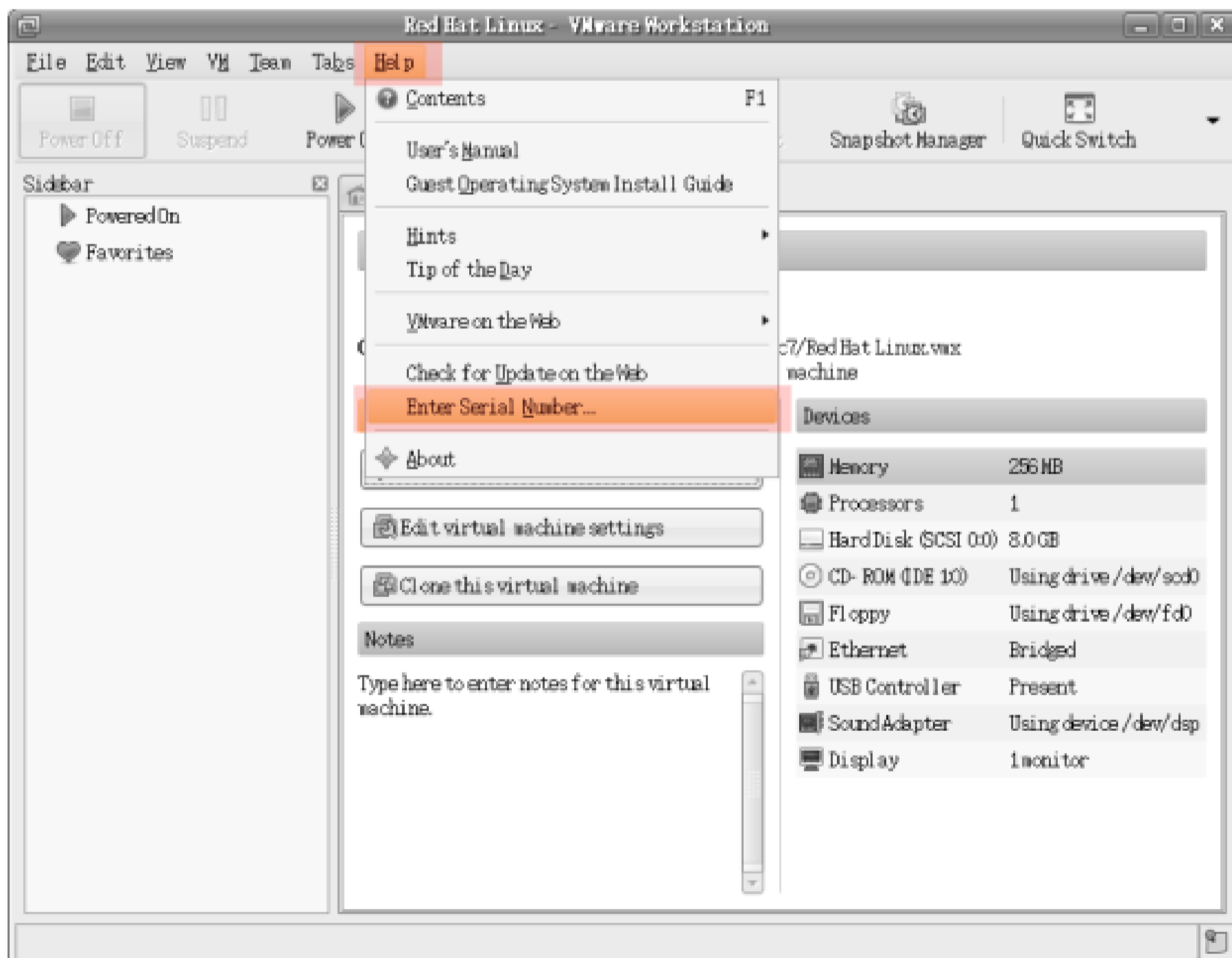
输入命令：**#tar zxvf vmware-fc7.tar.gz**

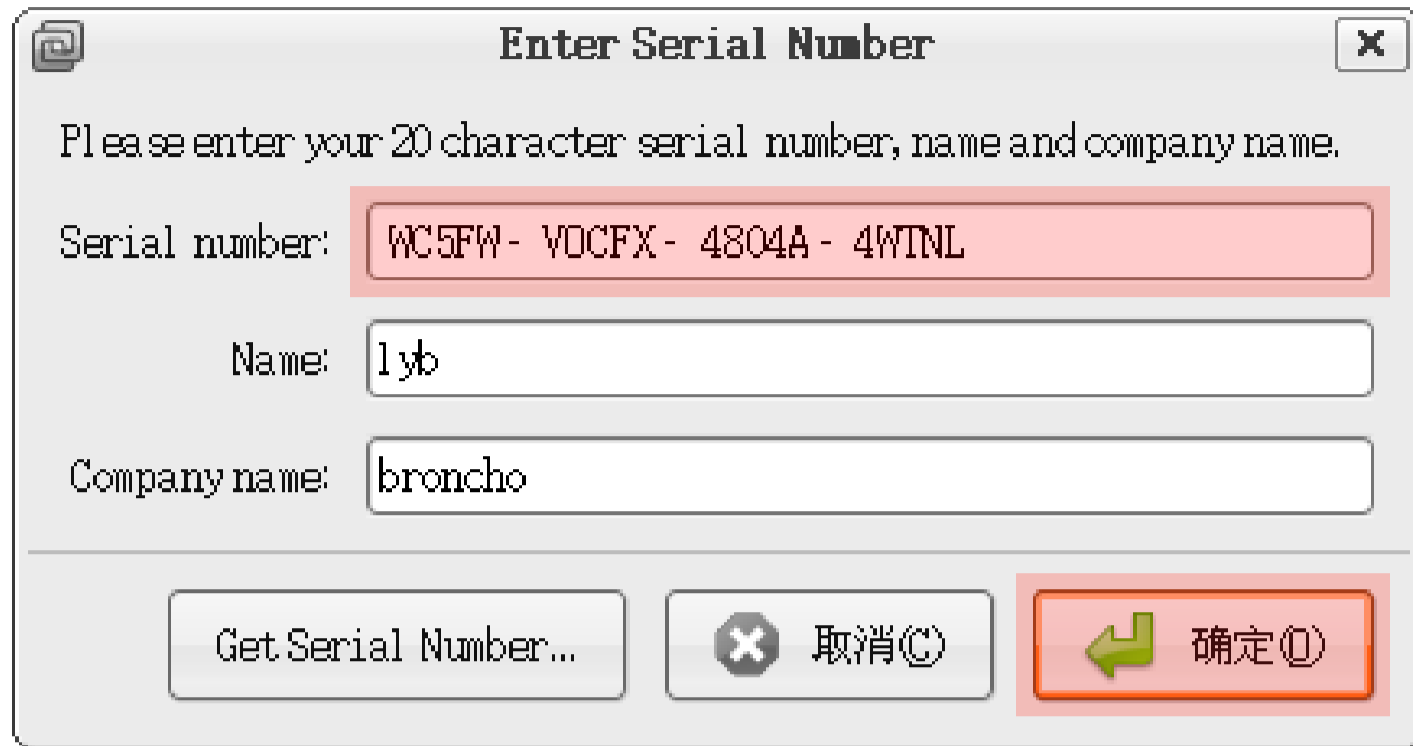


```
lyb@lyb-desktop: /media/sda7/software
lyb@lyb-desktop:/media/sda7/software$ ls -l vmware-fc7.tar.gz
-rw----- 1 lyb lyb 222622982 2008-03-27 09:43 vmware-fc7.tar.gz
lyb@lyb-desktop:/media/sda7/software$ tar zxvf vmware-fc7.tar.gz
```

3、注册 VMware

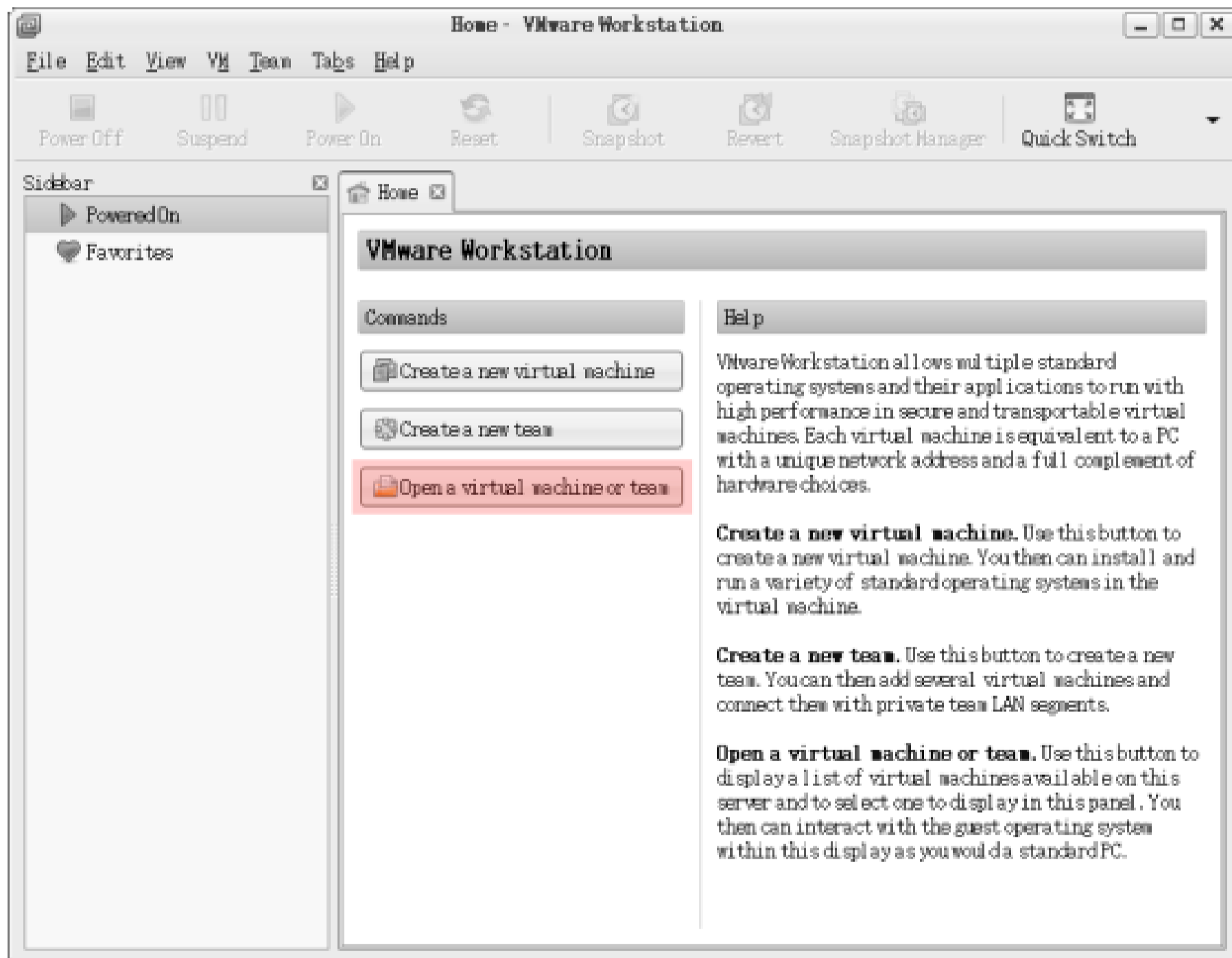
VMware 可以免费使用，但是需要注册码，在使用之前，我们先注册 VMware。

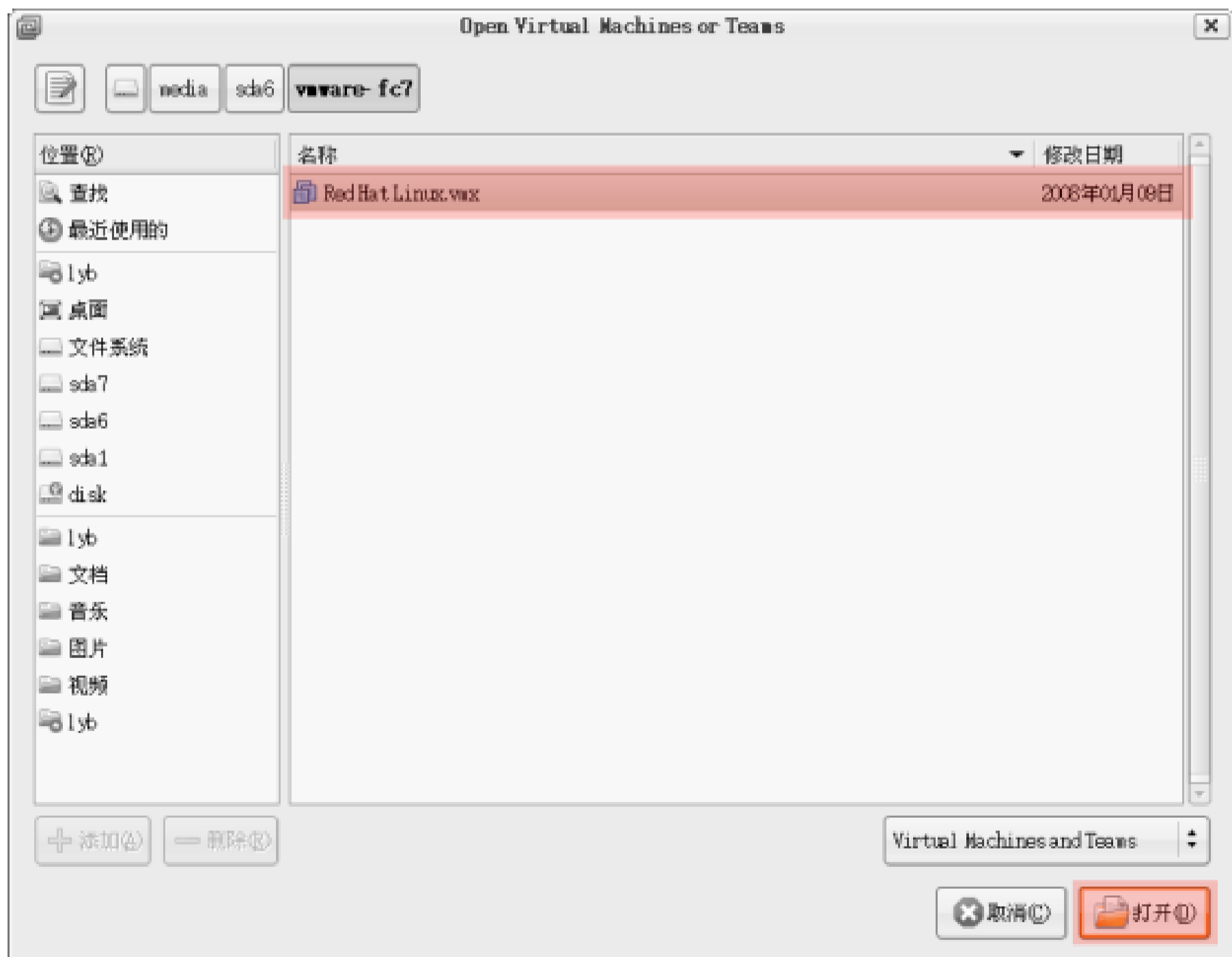
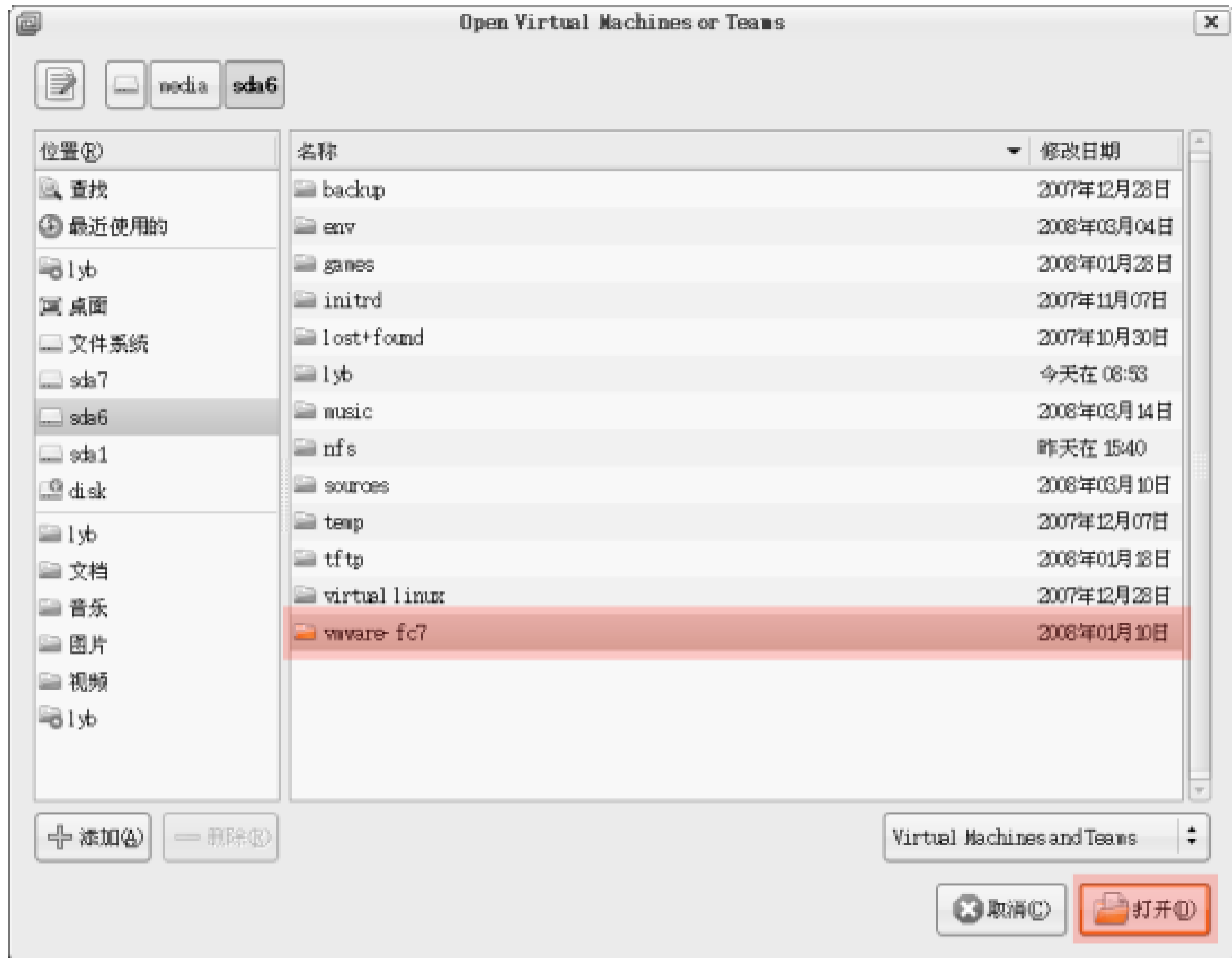




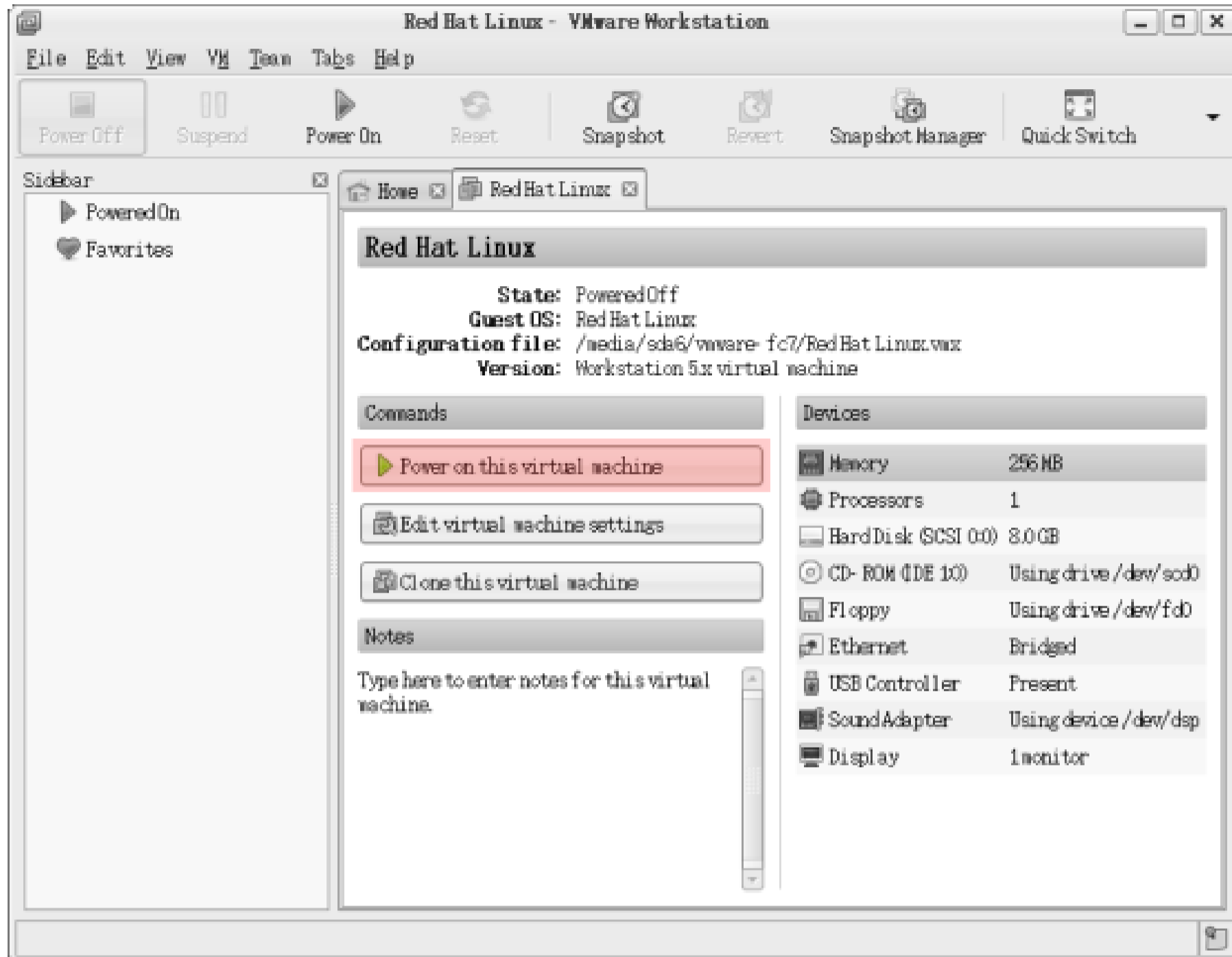
如何取得注册码，请上 <http://www.vmware.com> 了解详情。

3、导入我们的 vmware 镜像

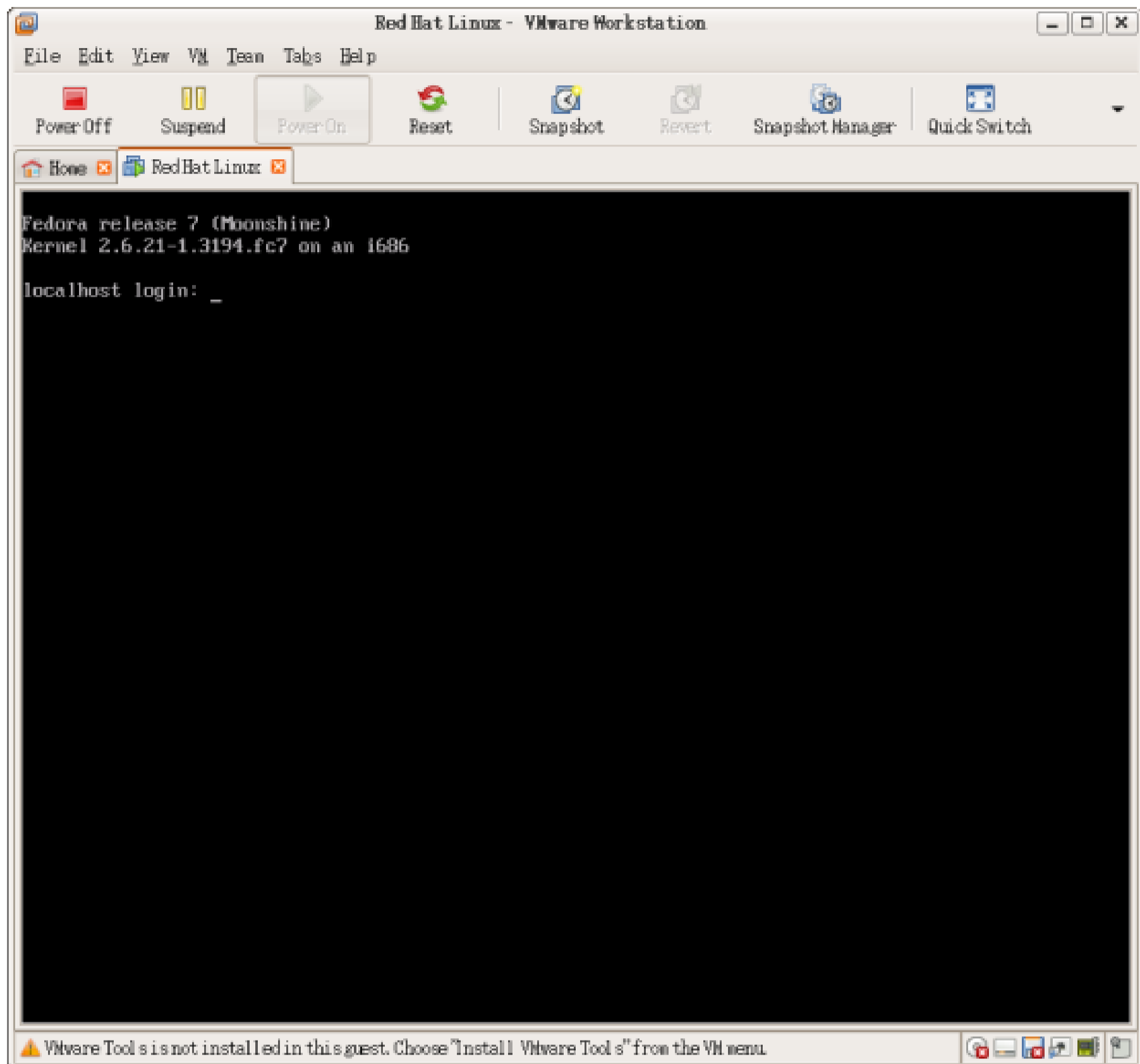




4、启动 fc7



看到下面这个画面，您已经成功安装了 **Broncho** 的开发虚拟机， 恭喜！



5、登录并查看虚拟机的 ip

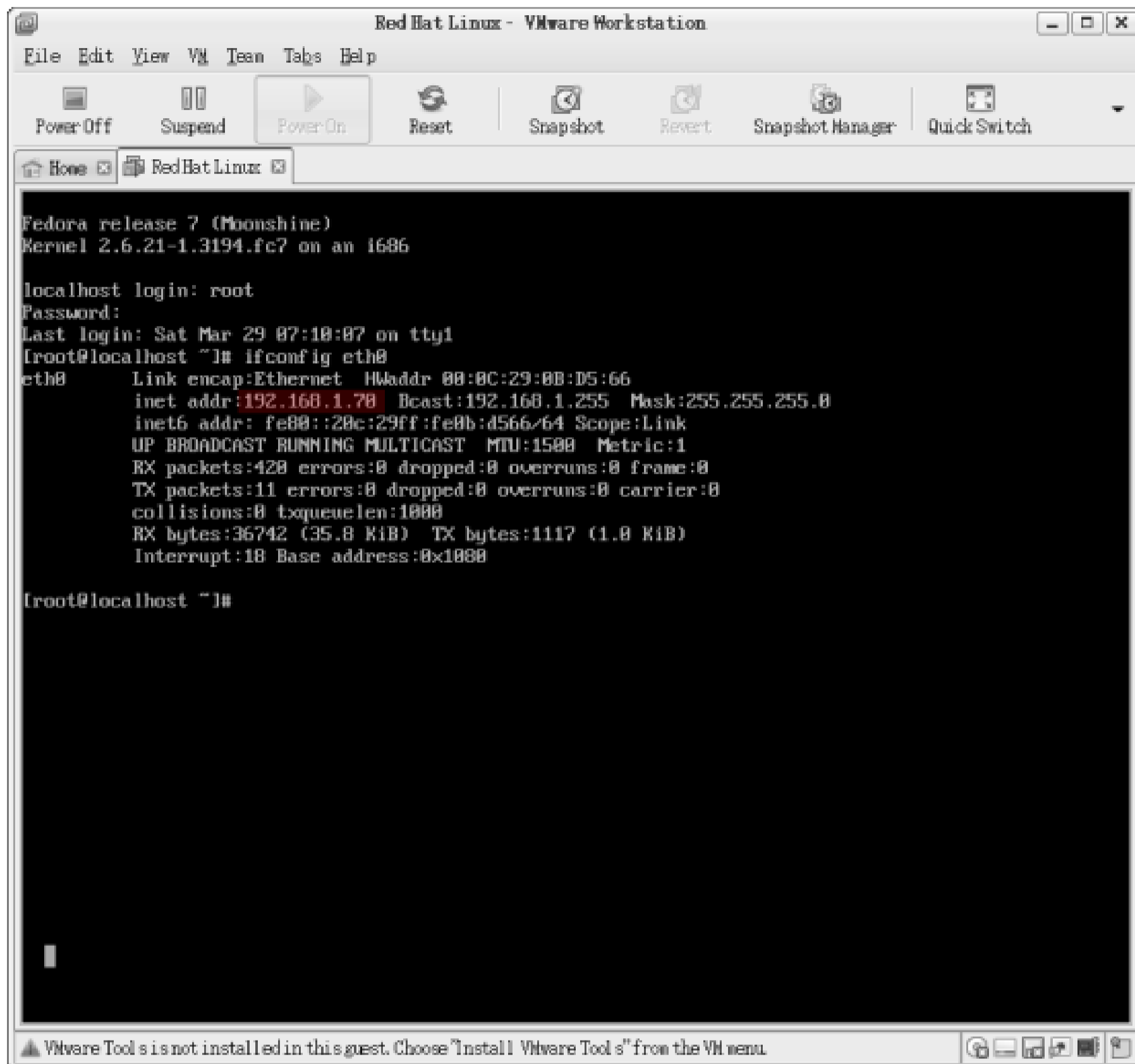
虚拟机中运行的就是 **Fedora core 7**，我们设定了初始的用户和密码，您可以通过初始用户和密码登录进虚拟机。

初始用户：**root** 初始密码：**root1023**

登录进虚拟后，您需要查看一下虚拟机的 **ip**，我们在后面用 **ssh** 登录的时候需要用到。

查看 **ip** 的命令是：**# ifconfig eth0**

虚拟机的 **ip** 都是通过 **dhcp** 服务器分配



红色方框里的就是你的 **ip** 地址（**192.168.1.70**），记住了，待会我们会用到。

6、通过 **ssh** 远程登录到虚拟机

🌟 打开一个 **shell** 终端，输入命令：`# ssh root@192.168.1.70`



```
root@localhost:~  
lyb@lyb-desktop:~$ ssh root@192.168.1.70  
root@192.168.1.70's password:  
Last login: Sat Mar 29 07:21:17 2008 from 192.168.1.87  
[root@localhost ~]#
```

以后我们编写程序、编译程序都通过终端登录到虚拟机里进行，虚拟机的另一个作用就是用来显示 **Broncho** 平台。

4.3 编写程序

开发环境配置好后，我们就可以开始着手写自己的 **Broncho** 应用程序。

现在打开一个终端，通过 **ssh** 登录到虚拟机上，使用 **vim** 将下面的例子输进去。并保存为 **testapp.c**。

一个简单的例子程序：

```
#include <gtk/gtk.h>

int main (int argc, char* argv[])
{
    GtkWidget* window = NULL;
    GtkWidget* label = NULL;

    gtk_init (&argc, &argv);

    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    label = gtk_label_new ("hello world");

    gtk_container_add (GTK_CONTAINER(window), label);

    gtk_widget_show_all (window);

    gtk_main ();
    return 0;
}
```

这个是一个非常简单的 **gtk** 界面应用，创建了一个窗口并显示“**hello world**”字符串。如果你之前有应用 **gtk** 的经验，你会发现这跟一般的 **gtk** 程序没什么区别。是的，得益于 **gtk** 和 **Directfb** 的优秀设计，基于 **gtk** 的程序，都能很容易的移植到 **broncho** 平台下。

 可用 **google** 查阅 **vim** 的教程

代码输进去后，我们再来创建一个 **Makefile** 文件，把下面的例子输进去，并保存为 **Makefile**。

一个很简单的 **Makefile**：

```
all: testapp.c
    gcc -o testapp testapp.c `pkg-config gtk+-2.0 --libs --cflags`

clean:
    rm -f *.o testapp
```

-  注意：`是按键 **1** 左边的符号，不是回车键左边的`
-  可用 **google** 查阅 **Makefile** 的相关知识
-  **testapp.c** 和 **Makefile** 放到同一目录


4.4 编译程序

4.3.1 环境变量设置

进入 **/broncho-sdk/work/build/broncho** 目录

输入命令： `# cd /broncho-sdk/work/build/broncho`

导入编译所需的环境变量


 输入命令： `# . pc_env.sh`

4.3.2 开始编译

进入 **testapp.c Makefile** 所在的目录，这里假如放在 **/root** 下

输入命令： `# cd /root`

编译您的第一个程序，用 **make** 命令

 输入命令： `# make`

确认程序编译成功

输入命令： `# ls -l testapp`

 运行 **make** 命令的时候，如果没有错误提示，一般都编译成功

```
root@localhost:~  
[root@localhost 1]# cd /root/  
[root@localhost ~]# make  
gcc -o testapp testapp.c `pkg-config gtk+-2.0 --libs --cflags`  
[root@localhost ~]# ls -l testapp  
-rwxr-xr-x 1 root root 6268 04-02 06:18 testapp  
[root@localhost ~]#
```

4.3.3 测试

新开一个终端，通过 **ssh** 登录到虚拟机上，我们测试自己的程序都在这个新开的终端里进行。将编辑代码的终端和调试程序的终端分开来，有利于您开发时的便利。我们可以称这个为运行 **Broncho** 平台的终端或者称为调试程序的终端。

进入 **/broncho-sdk/work/pemu/broncho/usr**

输入命令： `# cd /broncho-sdk/work/pemu/broncho/usr`

启动 **Broncho** 平台

🌟 输入命令： `# . pc_emu_run.sh`

运行后会出现一大堆信息，不要怕，不要理它，现在虚拟机才是你要关注的对象。看到下面这个界面了吗？没有的话，等等。



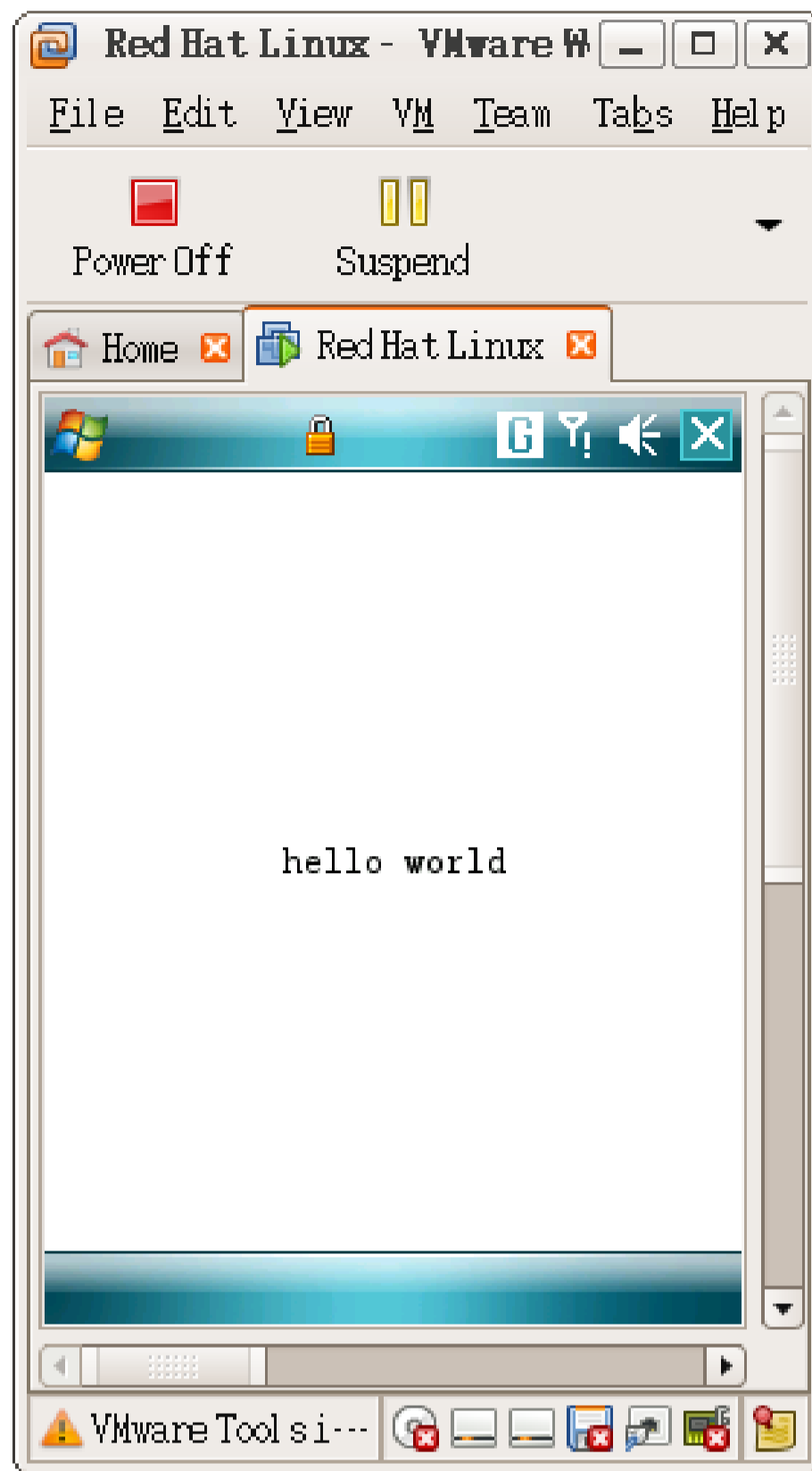
这就是我们 **Broncho** 平台的桌面，很像 **Windows mobile**? 那只是我们开发时的默认主题，内部完全不一样。

进到你编译 **testapp.c** 的目录

输入命令: `# cd /root`

运行编译好的 **testapp** 可执行文件

🌟 输入命令: `# ./testapp`



看到这个画面，恭喜您，您的第一个程序运行成功了！这个程序只显示“**hello world**”，除此之外什么也不做，但你实实在在的写出自己的第一个程序。如果没有跟预期的一样出现，不要泄气，回到第四章开头，重新再看看。

接下来的事情是休息下，回想下刚才所讲的内容，第四章我们讲了：

- 1、虚拟机 **Vmware** 的安装和使用。
- 2、如何通过虚拟机进行开发。
- 3、编写并编译了我们的第一个程序。
- 4、其中我们涉及到了 **ssh**、**vim**、**make**、**gcc** 等命令的使用

第五章带你继续深入学习开发 **Broncho** 平台的应用程序。

5.开发复杂的程序

第 4 章我们学习了开发环境的搭建和怎样编译一个简单的 **Broncho** 应用程序，这仅仅是局限于我们生成简单的程序。考虑到一个软件工程中程序代码达到成千上万、文件几十个、需要对软件国际化、存在几个程序相互依赖的时候、、、，这种做法并不能适应这些复杂的情况。下面我们学习使用优秀的 **GNU** 工具链：**autoscan**、**automake**、**autoconf**、**autoheader** 来管理我们的工程，并将工程融合到 **SDK**。

我们通过创建一个工程 **testapp** 来学习如何开发更复杂的程序。

5.1 建立一个软件工程

进到 **/broncho-sdk/work/app** 目录

输入命令： **# cd /broncho-sdk/work/app**

创建工程目录

🔥 输入命令： **# mkdir testapp**

输入命令： **# mkdir testapp/src**

在 **testapp/src** 里创建源文件 **testapp.c** 内容同第 4 章一样

在 **testapp/src** 目录里创建 **Makefile.am** 文件，这个文件就是 **automake** 工程相关的文件。内容如下：

```
bin_PROGRAMS=testapp

testapp_SOURCES=testapp.c

testapp_CFLAGS=@PACKAGE_CFLAGS@
testapp_LDFLAGS=@PACKAGE_LIBS@
```

在 **testapp** 目录里创建 **Makefile.am** 文件，内容如下：

```
SUBDIRS=src
```

在 **testapp** 目录里用 **autoscan** 命令生成基本的工程信息，**autoscan** 命令会在当前目录下生成一个 **configure.scan** 文件，需要将这个文件重命名 **configure.in**

🔥 输入命令： **# autoscan**

🔥 输入命令： **# mv configure.scan configure.in**

我们需要对 **configure.in** 进行一些修改，编辑它，让它内容如下：

```

#                                     -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ(2.61)
AC_INIT(configure.in)
AM_INIT_AUTOMAKE(testapp, 0.0.1)
AC_CONFIG_SRCDIR([src/testapp.c])
AC_CONFIG_HEADER([config.h])

# Checks for programs.
AC_PROG_CC

# Checks for libraries.
pkg_modules="gtk+-2.0"
PKG_CHECK_MODULES(PACKAGE, [$pkg_modules])
AC_SUBST(PACKAGE_CFLAGS)
AC_SUBST(PACKAGE_LIBS)

# Checks for header files.

# Checks for typedefs, structures, and compiler characteristics.





# Checks for library functions.

AC_CONFIG_FILES([Makefile
                 src/Makefile])
AC_OUTPUT


```

 对比一下 **configure.in** 修改前后的变化。


运行一系列命令，生成 **autoconf** 工程

-  输入命令: # aclocal
-  输入命令: # autoheader
-  输入命令: # automake -a -c --foreign
-  输入命令: # autoconf

测试一下是否成功

-  输入命令: # ./configure

运行后无任何出错提示，而且在当前目录和 **src** 目录产生了 **Makefile** 文件，说明配置成功。

 **Makefile** 是不是感觉似曾见过？在第 4 章出现过，猜猜它的用处。

清空工程文件，因为我们会在 **Broncho** 工程里面编译。

🔥 输入命令： `# make distclean`

5.2 将工程并入到 Broncho 工程

您已经成功配置好一个软件工程，接下来我们将它加到 **Broncho** 工程。

进到 `/broncho-sdk/work/build/broncho` 目录

输入命令： `# cd /broncho-sdk/work/build/broncho`

编辑 `packages.txt`，在文件的最后添加“**test app testapp**”，这相对应的就是“作者 所属类别 工程名”。

运行脚本 `update_generic_makefile.sh`，更新 **Broncho** 工程文件所需的文件。

🔥 输入命令： `# ./update_generic_makefile.sh`

编辑 `pc.mk`，在“`ALL_MODULES=`”后面添加“**testapp**”，改成如：

```
ALL_MODULES=testapp
```

5.3 Broncho 工程的使用

您已经修改好 **Broncho** 的工程文件，现在可以通过 **Broncho** 的工程文件来编译前面建好的工程。

进到 `/broncho-sdk/work/build/broncho` 目录

输入命令： `# cd /broncho-sdk/work/build/broncho`

导入环境变量

输入命令： `# . pc_env.sh`

📦 注意“.”和“`pc_env.sh`”之间的空格

开始编译

🔥 输入命令： `# make -f pc.mk MODULE=testapp`

如图，说明编译成功

```
root@localhost:/broncho- sdk/work/build/broncho
make[2]: Leaving directory `/broncho- sdk/work/app/testapp/i386'
make[2]: Entering directory `/broncho- sdk/work/app/testapp/i386'
Making install in src
make[3]: Entering directory `/broncho- sdk/work/app/testapp/i386/src'
make[4]: Entering directory `/broncho- sdk/work/app/testapp/i386/src'
test -z "/media/sdb1/work/pcemu/broncho/usr/local/bin" || /bin/mkdir -p "/media
/media/sdb1/work/pcemu/broncho/usr/local/bin"
/usr/bin/install -c 'testapp' '/media/sdb1/work/pcemu/broncho/usr/local/bin/t
estapp'
make[4]: Nothing to be done for `install-data-am'.
make[4]: Leaving directory `/broncho- sdk/work/app/testapp/i386/src'
make[3]: Leaving directory `/broncho- sdk/work/app/testapp/i386/src'
make[3]: Entering directory `/broncho- sdk/work/app/testapp/i386'
make[4]: Entering directory `/broncho- sdk/work/app/testapp/i386'
make[4]: Nothing to be done for `install-exec-am'.
make[4]: Nothing to be done for `install-data-am'.
make[4]: Leaving directory `/broncho- sdk/work/app/testapp/i386'
make[3]: Leaving directory `/broncho- sdk/work/app/testapp/i386'
make[2]: Leaving directory `/broncho- sdk/work/app/testapp/i386'
make[1]: Leaving directory `/broncho- sdk/work/build/broncho'
package(s) is all done
[root@localhost broncho]#
```

切换到你运行 **broncho** 平台的那个终端，运行一下，看是否成功。

🌟 输入命令： `# testapp`

如果出现更第 4 章一样的“**hello world**”界面的话，证明工程编译成功！如果没有跟预期的一样，肯定是中途漏了点什么，回到第 5 章开头重新看看。

在第 5 章我们初步的接触了：

- 1、利用 **GNU** 工具链创建 **automake** 工程。
- 2、将工程添加到 **Broncho** 工程里。

通过第 5 章的学习，您已经是开发人员了，接下来会系统的接触一些更深的内容。

6.Linux 系统使用基础

6.1 基本命令

表示是 **root** 的 **shell**.\$表示一般用户的 **shell**

1.man 对你熟悉或不熟悉的命令提供帮助解释

eg:man ls 就可以查看 **ls** 相关的用法

注：按 **q** 键或者 **ctrl+c** 退出,在 **linux** 下可以使用 **ctrl+c** 终止当前程序运行

2.ls 查看目录或者文件的属*,列举出任一目录下面的文件

eg: ls /usr/man

ls -l

a.d 表示目录(**directory**),如果是一个"**-**"表示是文件,如果是 **l** 则表示是一个连接文件(**link**)

b.表示文件或者目录许可权限.分别用可读(**r**),可写(**w**),可运行(**x**).

3.cp 拷贝文件

eg: cp filename1 filename2 //把 **filename1** 拷贝成 **filename2**

cp 1.c netseek/2.c //将 **1.c** 拷到 **netseek** 目录下命名为 **2.c**

4.rm 删除文件和目录

eg: rm 1.c //将 **1.c** 这个文件删除

5.mv 移走目录或者改文件名

eg: mv filename1 filename2 //将 **filename1** 改名为 **filename2**

mv qib.tgz ../qib.tgz //移到上一级目录

6.cd 改变当前目录 **pwd** 查看当前所在目录完整路径

eg: pwd //查看当前所在目录路径

cd netseek //进入 **netseek** 这个目录

cd //退出当前目录

7.cat,more 命令

将某个文件的内容显示出来.两个命令所不同的是:**cat** 把文件内容一直打印出来,而 **more** 则分屏显示

eg: cat > 1.c //就可以把代码粘帖到 **1.c** 文件里,按 **ctrl+d** 保存代码。

cat 1.c 或 **more 1.c** //都可以查看里面的内容。

gcc -o 1 1.c //将 **1.c** 编译成 **.exe** 文件,我们可以用此命编译出代码

8.chmod 命令 权限修改 用法: **chmod** 一位 **8** 进制数 **filename**

eg: chmod u+x filename //只想给自己运行,别人只能读

//**u** 表示文件主人, **g** 表示文件文件所在组. **o** 表示其他人 ;**r** 表可读,**w** 表可写,**x** 表可以运行

chmod g+x filename //同组的人来执行

9. clear,date 命令 **clear**:清屏,相当与 **DOS** 下的 **cls**;**date**:显示当前时间.

10.mount 加载一个硬件设备

用法:**mount** [参数] 要加载的设备 载入点

eg: mount /dev/cdrom

cd /mnt/cdrom //进入光盘目录

11.su 在不退出登陆的情况下,切换到另外一个人的身份

用法: **su -l** 用户名(如果用户名缺省,则切换到 **root** 状态)

eg:su -l netseek (切换到 **netseek** 这个用户,将提示输入密码)

12. whoami, whereis, which, id

//whoami:确认自己身份.

//whereis:查询命令所在目录以及帮助文档所在目录.

//which:查询该命令所在目录(类似 whereis)

//id:打印出自己的UID以及GID.(UID:用户身份唯一标识.GID:用户组身份唯一标识.每一个用户只能有一个唯一的UID和GID.)

eg: whoami //显示你自己登陆的用户名

whereis bin 显示 bin 所在的目录, 将显示为: /usr/local/bin

which bin

13. grep, find grep:文本内容搜索;find:文件或者目录名以及权限属主等匹配搜索

eg: grep success * /*查找当前目录下面所有文件里面含有 success 字符的文件

14.kill 可以杀死某个正在进行或者已经是 dest 状态的进程

eg; ps ax

15.passwd 可以设置口令

16.history 用户用过的命令

eg: history //可以显示用户过去使用的命令

17.!! 执行最近一次的命令

18.mkdir 命令

eg: mkdir netseek //创建 netseek 这个目录

19.tar 解压命令

eg: tar -zxvf nmap-3.45.tgz //将这个解压到 nmap-3.45 这个目录里

20.finger 可以让使用者查询一些其他使用者的资料

eg: finger //查看所用用户的使用资料

finger root //查看 root 的资料

6.2 GNU 编译工具

7. Broncho 工程详解

暂略

8.附录

 文档中的程序，都可以从/**broncho-sdk/example** 找到。或者可以从我们的网站上下载，下载地址是：

<http://www.broncho.cn/download/sdk/examples.tar.gz>。

参考资料：

<http://www.broncho.cn> **Broncho** 的官方网站，欢迎您访问。

<http://www.gtk.org> **gtk** 的官方网站，很多 **gtk** 文档资料。

<http://www.gnome.org> **gnome** 的官方网站，**gnome** 是架构于 **gtk** 的桌面系统。

<http://www.gnu.org> **gnu** 的官方网站，很多优秀的 **gnu** 软件和文档。

<http://www.freedesktop.org> 自由桌面官方网站，有很多桌面标准资料。

<http://www.linux.org> **linux** 内核官方网站。

<http://www.vim.org> **vim** 的官方网站。