

实验一 嵌入式开发环境构建

实验指导

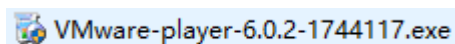
参考指导: 嵌入式 Linux 移植应用驱动-标准版.pdf 第 1 章, 第 2 章, 第 3 章, 第 6 章

说明: 华清远见开发环境是基于 Ubuntu 12.04 LTS 64-bit 操作系统搭建的, 使用 VMware Player 免费版作为虚拟机工具软件。用作 Linux 和 Android 的编译与开发。(有两个版本的开发环境 12.04 和 14.04, 可以自行选择, 我做的是 14.04)

1. 安装 VMware Player 虚拟机软件

VMware Player 从 6.0 版本之后默认支持中文, 所以华清远见开发环境 V12B 使用当前最新版的 VMware Player (版本号为 6.0.2 build-1744117), 如要正常使用此开发环境, 必须保证 VMware Player 版本号大于等于当前给出的版本号, 否则可能会出现因为 VMware Tools 版本过高引起虚拟机无法正常启动的情况。

打开 (实验\实验一\VMware Player) 目录下的 VMware Player 安装程序。



根据安装向导, 按照默认项进行安装。

2. 运行开发环境

2.1 解压虚拟镜像

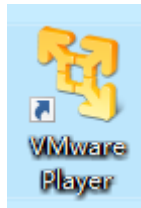
打开 (实验\实验一\华清远见开发环境 V14A), 解压镜像。

Ubuntu_14.04_64-bit_farsight.7z

可以使用 (实验\实验一\7zip) 7zip进行解压, 也可以winrar等。

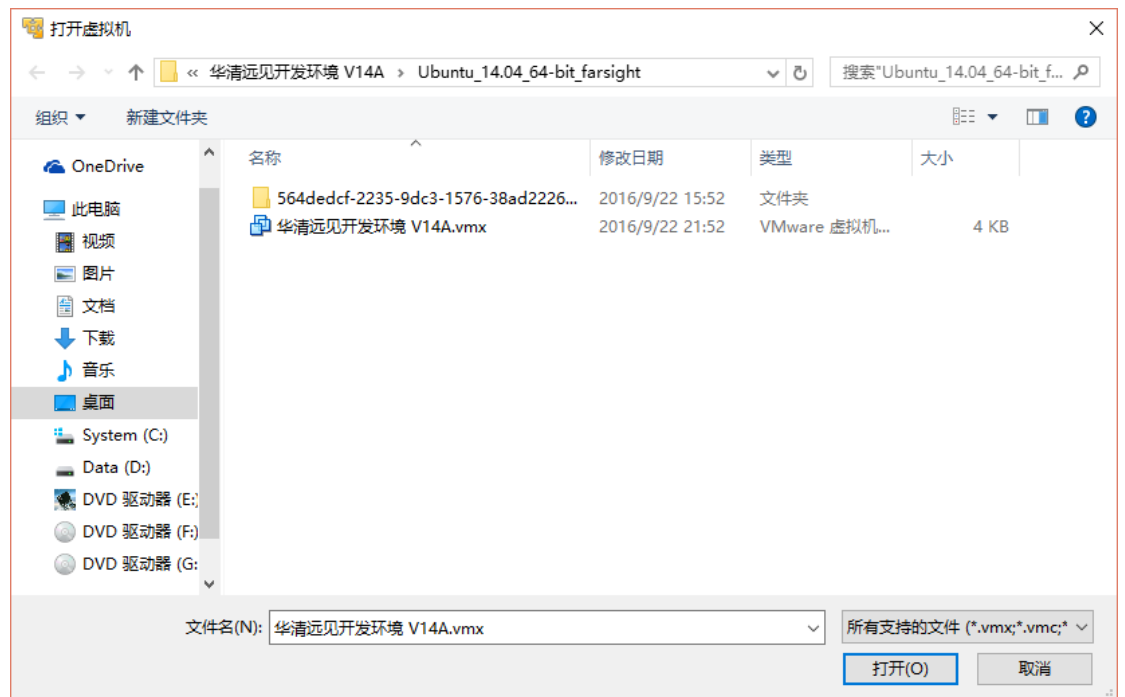
2.2打开虚拟机

打开VMware Player



打开虚拟镜像, 在解压的镜像文件中找到“华清远见开发环境V14A”





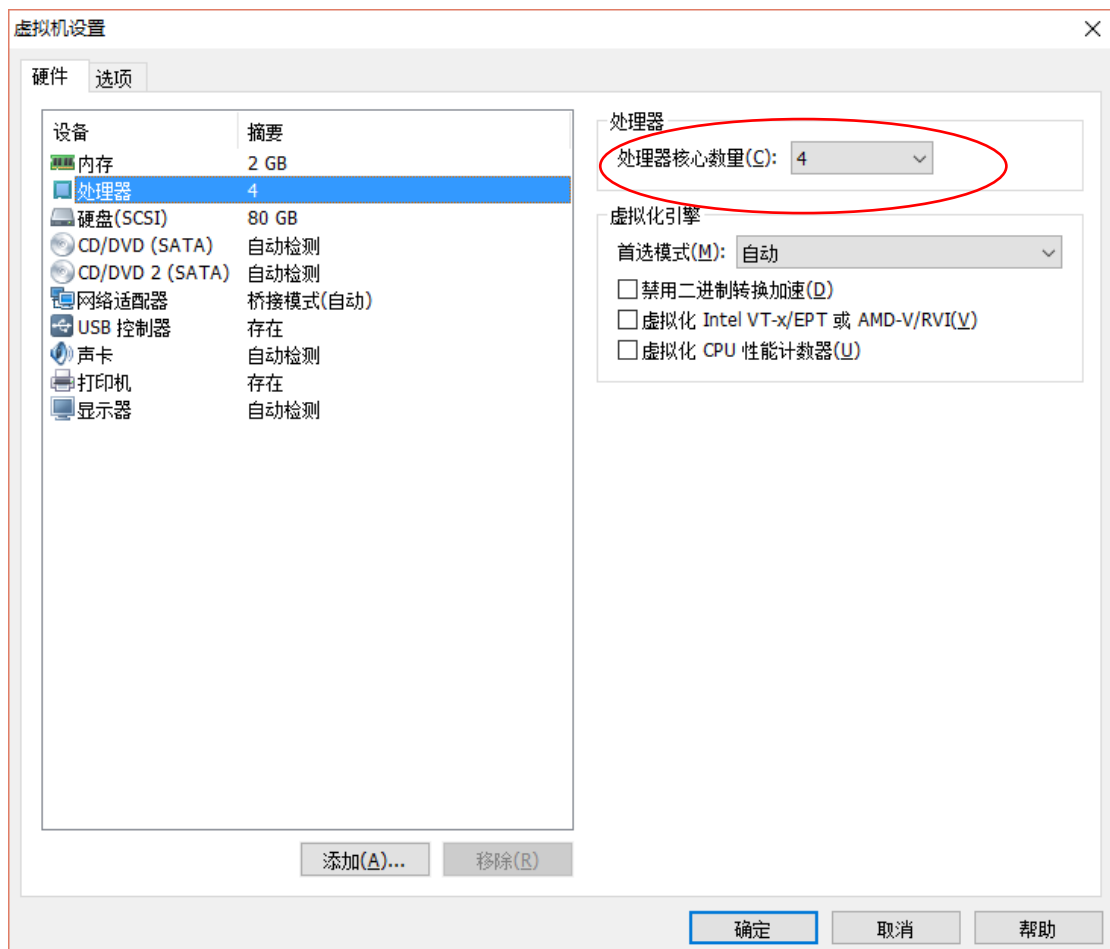
配置优化虚拟机



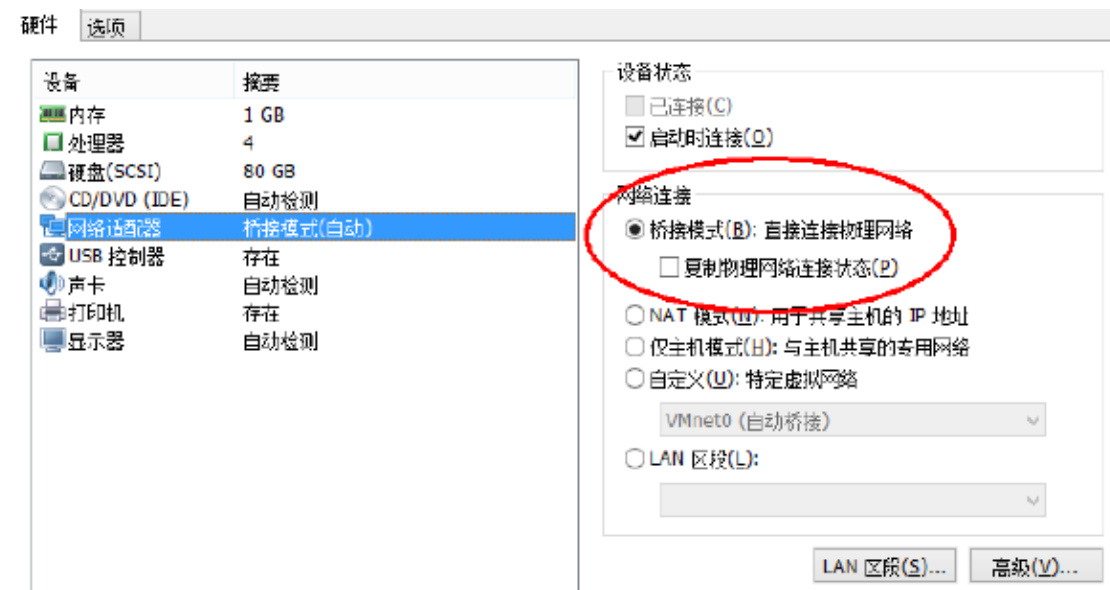
增加内存到1G。



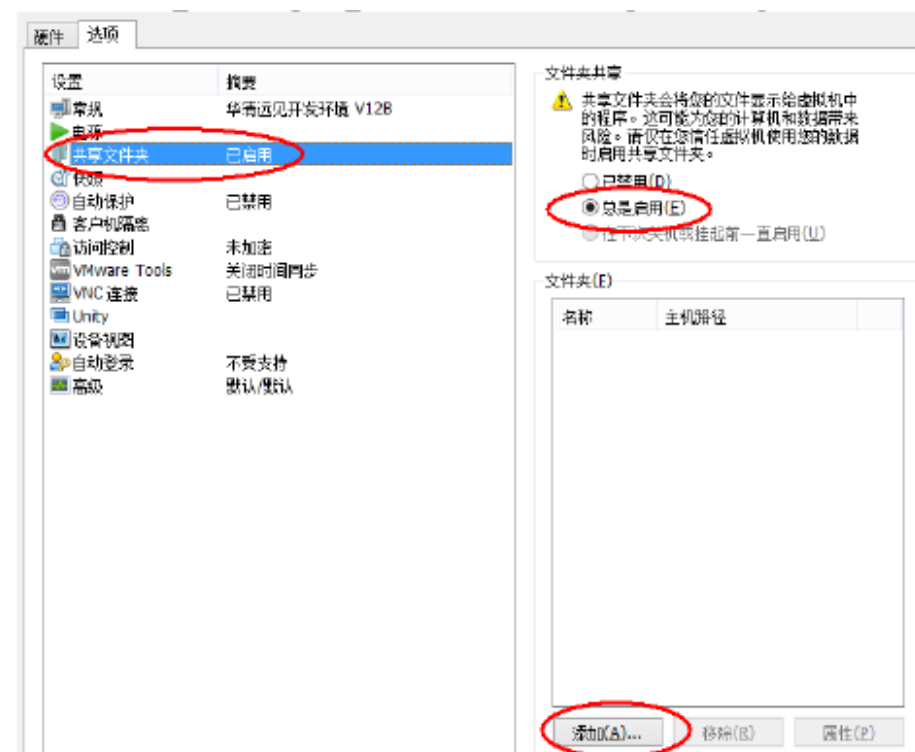
处理器为4核（设置的总核心数不要超过**CPU** 的核心数）。



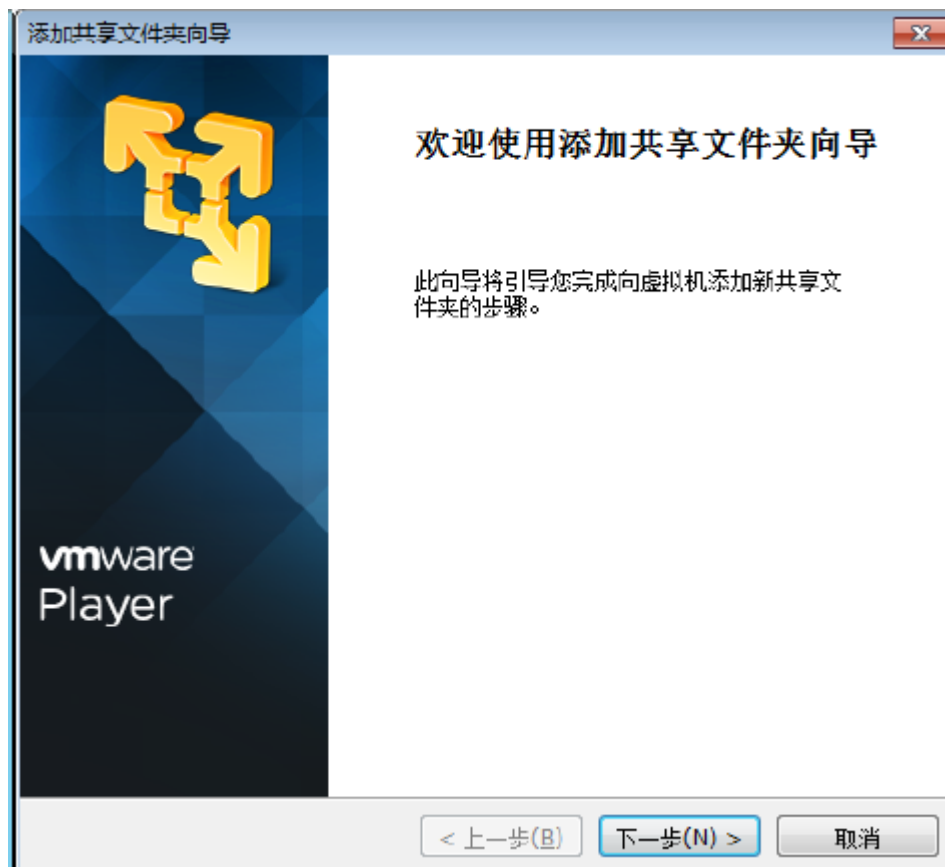
确认网络连接为桥接模式。



增加共享目录

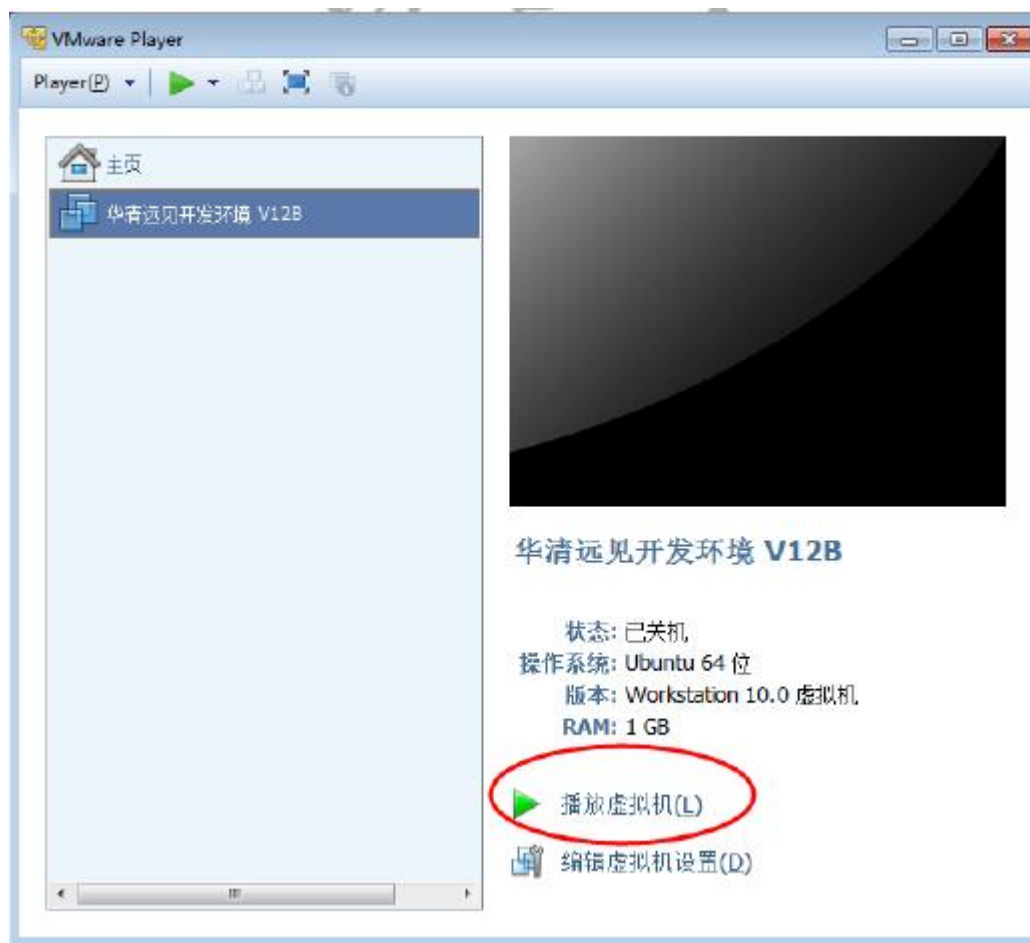


点击添加根据向导添加虚拟机与PC共享目录

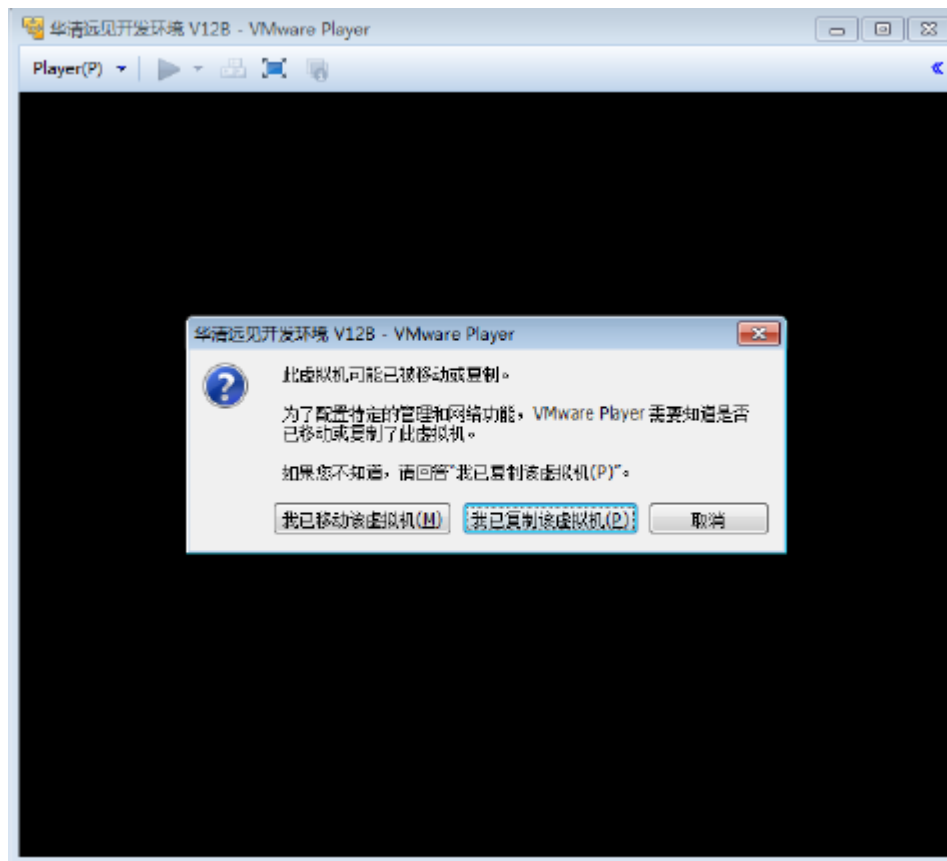




启动虚拟机

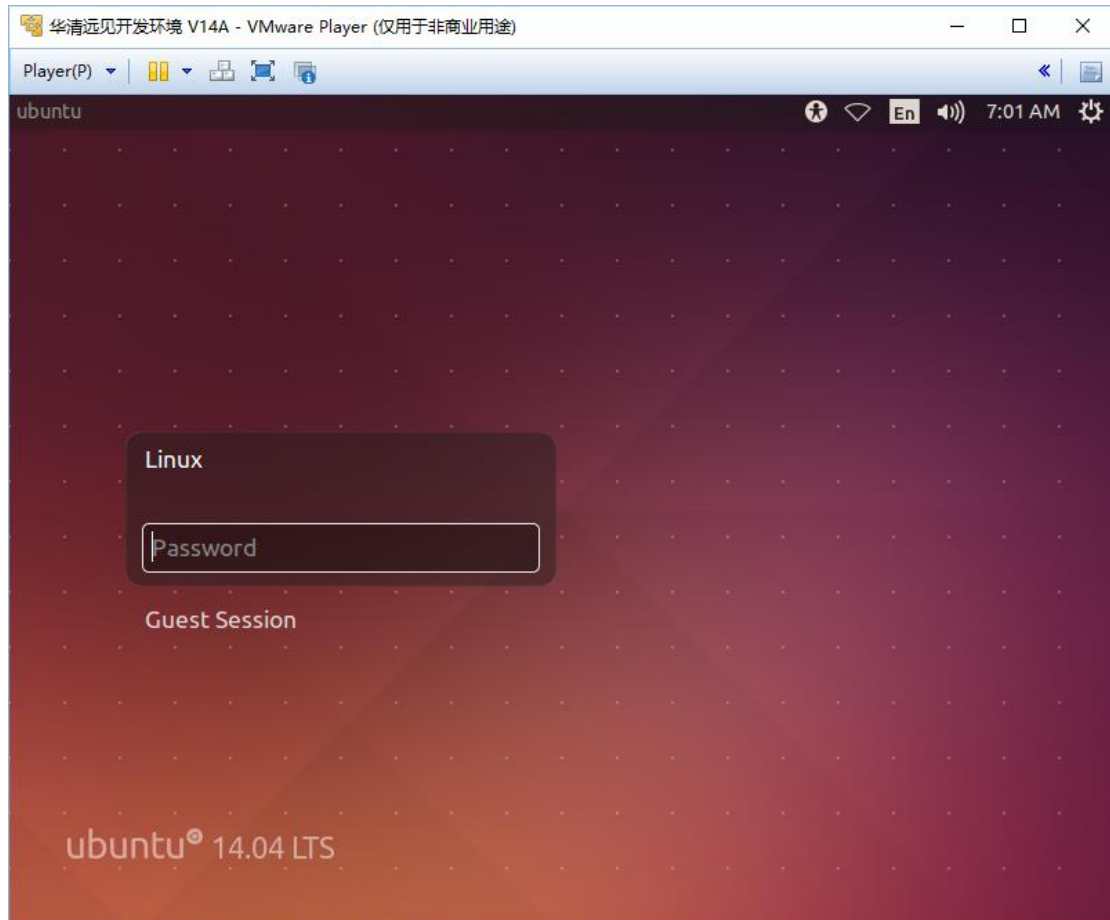


选择我已复制虚拟机



可能遇到的问题：此主机支持 Intel VT-x,但 Intel VT-x 处于禁用状态。启动虚拟机时，boot 中 cpu configuration 中 Intel Virtualization Technology 未开启。

用户名linux，密码默认为1.



3. 嵌入式Linux主机调试环境搭建

3.1 Linux系统配置TFTP

华清远见开发环境中已经包含tftp服务，安装不必操作，有意者可以搜索相关资料自行安装，在此，仅测试tftp功能。

```
$ cd /tftpboot
```

```
$ ls
```

```
root@ubuntu: /tftpboot
root@ubuntu:/# cd /tftpboot/
root@ubuntu:/tftpboot# cat test
this is a test file!

root@ubuntu:/tftpboot#
```

可以自己建一个test文件，里面内容为this is a file!

\$ cd ~ //Linux 下波浪线【~】代表用户的home 目录，我们俗称主目录或者家目录

\$ tftp 127.0.0.1

> get test

>q

test从/tftpboot目录传输到~目录。

```
root@ubuntu:/tftpboot# cd ~
root@ubuntu:~# tftp 127.0.0.1
tftp> get test
tftp> q
root@ubuntu:~# ls
test  workdir
root@ubuntu:~# cat test
this is a test file!

root@ubuntu:~#
```

```
root@ubuntu:/# cd ~
root@ubuntu:~# tftp localhost
tftp> get test
tftp> q
root@ubuntu:~# ls
test
root@ubuntu:~# cat test
this is a test file!
```

可能遇到的问题: tftp transfer timed out!

配置# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"

TFTP_DIRECTORY="/tftpboot"

TFTP_ADDRESS="0.0.0.0:69"

TFTP_OPTIONS="-l -c -s"

tftpboot 文件夹的权限 `chmod 777 tftpboot`

重启服务 `service tftpd-hpa restart`

3.2 linux 系统配置 NFS

\$ `sudo vim /etc/exports`

NFS 允许挂载的目录及权限在文件 `/etc/exports` 中进行了定义。例

如, 我们要将 `/source/rootfs` 目录共享出来, 那么我们需要在

`/etc/exports` 文件末尾添加如下一行:

`/source/rootfs`

`*(rw,sync,no_root_squash,no_subtree_check)`

```
root@ubuntu: ~  
# /etc/exports: the access control list for filesystems which may be exported  
# to NFS clients.  See exports(5).  
#  
# Example for NFSv2 and NFSv3:  
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)  
#  
# Example for NFSv4:  
# /srv/nfs4       gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)  
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)  
#  
/source/rootfs *(rw,sync,no_root_squash,no_subtree_check)  
~  
~
```

其中：/source/rootfs 是要共享的目录，*代表允许所有的网络段访问，rw 是可读写权限,sync 是资料同步写入内存和硬盘，no_root_squash 是NFS 客户端分享目录使用者的权限，如果客户端使用的是root 用户，那么对于该共享目录而言，该客户端就具有root 权限。

重启服务：

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

重启服务成功如下显示。(如果设置的路径没有相应内容，会提示错误，可以先忽略这个问题)

4. 交叉开发环境搭建

查看 ubuntu 的 ip `ifconfig -a`

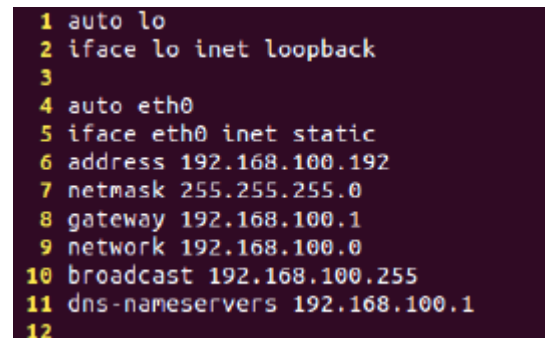
4.1虚拟机网络方式为桥接模式，此状态下虚拟机下的操作系统和主机操作系统为平级状态。为了调试方便，我们可以给虚拟机下的Ubuntu 一个静态的IP 地址。

假设我们使用的网络地址为192.168.100.x 段的，那么我可以给Ubuntu 分配一个IP 为192.168.100.192， 配置过程如下所示。

配置虚拟机网络环境。

```
$ sudo vim /etc/network/interfaces
```

修改文件如下图所示。保存退出。



```
1 auto lo
2 iface lo inet loopback
3
4 auto eth0
5 iface eth0 inet static
6 address 192.168.100.192
7 netmask 255.255.255.0
8 gateway 192.168.100.1
9 network 192.168.100.0
10 broadcast 192.168.100.255
11 dns-nameservers 192.168.100.1
12
```

```
$ sudo /etc/init.d/networking restart
```

使用【ifconfig】命令查看我们修改的结果。如果没有修改成功，重复上述步骤，或者重新启动虚拟机 Ubuntu 系统即可。

4.2 配置交叉工具链

华清远见开发环境包含了 3 个版本的交叉工具链，路径在 /usr/local/toolchain/下（旧版本在/home/linux/toolchain/下），不必再次解压，直接指向路径即可。下列步骤均按照新版本的路径介绍，旧版本的用户可以选择更新或者自行修改路径即可。

一般情况下，当我们输入一个 Linux 命令，比如【ls】可以直接执行功能，但我当前的目录并没有【ls】这个文件。【ls】这个命令文件在【/bin】这个目录中，我们可以执行这是因为这些命令所在的路径包含在了用户的环境变量中。

修改文件 ~/.bashrc，添加如下内容

```
$ vim ~/.bashrc // 注意 bashrc 前面有句点
```

在文件末尾添加下一行

```
export PATH=$PATH:/usr/local/toolchain/toolchain-4.6.4/bin/
```

重启配置文件

```
$ source ~/.bashrc
```

工具链的测试

```
$ arm-none-linux-gnueabi-gcc -v
```

```
root@ubuntu:~# arm-none-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-none-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/usr/local/toolchain/toolchain-4.6.4/bin/../libexec/gcc/arm-arm1176jzfssf-lin
ux-gnueabi/4.6.4/lto-wrapper
Target: arm-arm1176jzfssf-linux-gnueabi
Configured with: /work/builddir/src/gcc-4.6.4/configure --build=i686-build_pc-linux-gnu --host=i6
86-build_pc-linux-gnu --target=arm-arm1176jzfssf-linux-gnueabi --prefix=/opt/TuxamitoSoftToolchai
ns/arm-arm1176jzfssf-linux-gnueabi/gcc-4.6.4 --with-sysroot=/opt/TuxamitoSoftToolchains/arm-arm11
76jzfssf-linux-gnueabi/gcc-4.6.4/arm-arm1176jzfssf-linux-gnueabi/sysroot --enable-languages=c,c++
--with-arch=armv6zk --with-cpu=arm1176jzf-s --with-tune=arm1176jzf-s --with-fpu=vfp --with-float
=softfp --with-pkgversion='crosstool-NG hg+default-2685dfa9de14 - tc0002' --disable-sjlj-exceptio
ns --enable-__cxa_atexit --disable-libmudflap --disable-libgomp --disable-libssp --disable-libqua
dmath --disable-libquadmath-support --with-gmp=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/bui
ldtools --with-mpfr=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-mpc=/work/bu
ilddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-ppl=/work/builddir/arm-arm1176jzfssf-lin
ux-gnueabi/buildtools --with-cloog=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --wi
th-libelf=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-host-libstdcxx='-stati
c-libgcc -WL,-Bstatic,-lstdc++,-Bdynamic -lm' --enable-threads=posix --enable-target-optspace --w
ithout-long-double-128 --disable-nls --disable-multilib --with-local-prefix=/opt/TuxamitoSoftTool
chains/arm-arm1176jzfssf-linux-gnueabi/gcc-4.6.4/arm-arm1176jzfssf-linux-gnueabi/sysroot --enable
-c99 --enable-long-long
Thread model: posix
gcc version 4.6.4 (crosstool-NG hg+default-2685dfa9de14 - tc0002)
```

5. GCC 编译测试

GNU CC (简称为 Gcc) 是 GNU 项目中符合 ANSI C 标准的编译系统，能够编译用 C、C++ 和 Object C 等语言编写的程序。gcc 不仅功能强大，而且可以编译如 C、C++、Object C、Java、Fortran、Pascal、Modula-3 和 Ada 等多种语言，而且 gcc 又是一个交叉平台编译器，它能够在当前 CPU 平台上为多种不同体系结构的硬件平台开发软件，因此尤其适合在嵌入式领域的开发编译。

通过实验学习 gcc 编译器编译c 程序的方法, 熟悉gcc 编译程序的各个阶段。

5.1 建立相关目录

```
$ cd workdir/linux/application
```

```
$ mkdir 6-gcc
```

将 helloworld.c 文件拷贝到 share 文件夹下。

5.2 编译代码

```
$ arm-none-linux-gnueabi-gcc helloworld.c -o hello
```

```
$ mkdir /source/rootfs/app
```

```
$ cp hello /source/rootfs/app/
```

5.3 执行代码

```
./hello
```

```
$ arm-none-linux-gnueabi-gcc helloworld.c -o hello
```

```
$ mkdir /source/rootfs/app
```

```
$ cp hello /
```

查看生成文件类型 file hello 为二进制文件。

```
root@ubuntu:/source/rootfs/app# file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.38, not stripped
```