

一、交叉编译环境的搭建

1、 介绍： 最终的程序 --- 虚拟机上编译， arm 开发板上运行。

交叉编译工具： arm-linux-gcc

我们开发使用的编译器为 arm-linux-gcc， 我们使用的为 4.5.1 版本。解压到根目录并把生成的 bin 路径加载到环境变量当中即可。

2、 安装流程

拷贝安装包到虚拟机里面；解压到根目录 tar -xvf arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz -C / ；

设置环境变量 export PATH=\$PATH:/opt/FriendlyARM/toolschain/4.5.1/bin ；

添加到开机自启动的文件里面 ~/.bashrc ；

重启或者 source ~/.bashrc ；

命令： arm-linux-gcc -v //打印安装的交叉编译工具的版本号

3、 编译器类似于 gcc -E、 -S、 -o ；

二、uboot 编译烧写

1、 把 uboot_tiny4412-sdk1506.tar.bz2 拷贝到虚拟机里面解压， 并进入解压好的目录。

```
[root@localhost ~]# tar -xvf uboot_tiny4412-sdk1506.tar.bz2
```

```
[root@localhost ~]# cd uboot_tiny4412-sdk1506
[root@localhost uboot_tiny4412-sdk1506]# ls
api          CREDITS     MAINTAINERS  README
arch         disk        MAKEALL      rules.mk
board        doc         Makefile     sd_fuse
boards.cfg   drivers     mkconfig     snapshot.commit
common       examples    nand_spl     tools
config.mk    fs          net          uboot_tiny4412-sdk1506.tar.bz2
COPYING      include     onenand_ipl
COPYING.txt  lib         post
```

2、 执行 #make tiny4412_config

```
[root@localhost uboot_tiny4412-sdk1506]# make tiny4412_config
awk '(NF && $1 !~ /^#/)' { print $1 " : " $1 "_config; $(MAKE)" }' boards.cfg > .
boards.depend
Configuring for tiny4412 board...
```

3、执行 `#make` ；最后会生成一个 `u-boot.bin` 的二进制文件。如下：

```
none-linux-gnu-eabi/4.5.1 -lgcc -Map u-boot.map -o u-boot
arm-linux-ld: warning: creating a DT_TEXTREL in object.
arm-linux-objcopy -O srec u-boot u-boot.srec
arm-linux-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
[root@localhost uboot_tiny4412-sdk1506]#
```

4、执行 `#cp ./tools/mkimage /bin`

```
[root@localhost uboot_tiny4412-sdk1506]# cp ./tools/mkimage /bin
```

5、uboot 烧写到 SD 卡。

A、在 windows 在制作一个启动盘(就是利用 SD-Flasher.exe 从新格式化然后分区)然后在你的虚拟机软件上方，选择虚拟机。可移动设备，将 SD 卡挂接到虚拟机中。然后终端执行 `#ls /dev/sd*`，发现 `sdb`(或 `sdc`)是 sd 卡的挂接点。

```
[root@localhost ~]# ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sdb  /dev/sdb1
```

B、然后跳到解压好的 uboot 文件夹里的 `sd_fuse` 里面。执行 `make`

```
[root@localhost uboot_tiny4412-sdk1506]# cd sd_fuse/
[root@localhost sd_fuse]# make
gcc -o mkb12 V310-EVT1-mkb12.c
gcc -o sd_fdisk sd_fdisk.c
[root@localhost sd_fuse]#
```

C、然后跳转到 `tiny4412` 文件夹里面。执行 `#!/sd_fusing.sh /dev/sdb` 直到 uboot 编译烧写完成。

```
[root@localhost tiny4412]# ./sd_fusing.sh /dev/sdb
```

三、内核编译烧写 (配置内核)

1、将 `linux-3.5-20140822.tgz` 文件拷贝到虚拟机解压

```
[root@localhost ~]# tar -xvf linux-3.5-20140822.tgz
```

2、跳转到解压好的目录：`linux-3.5`

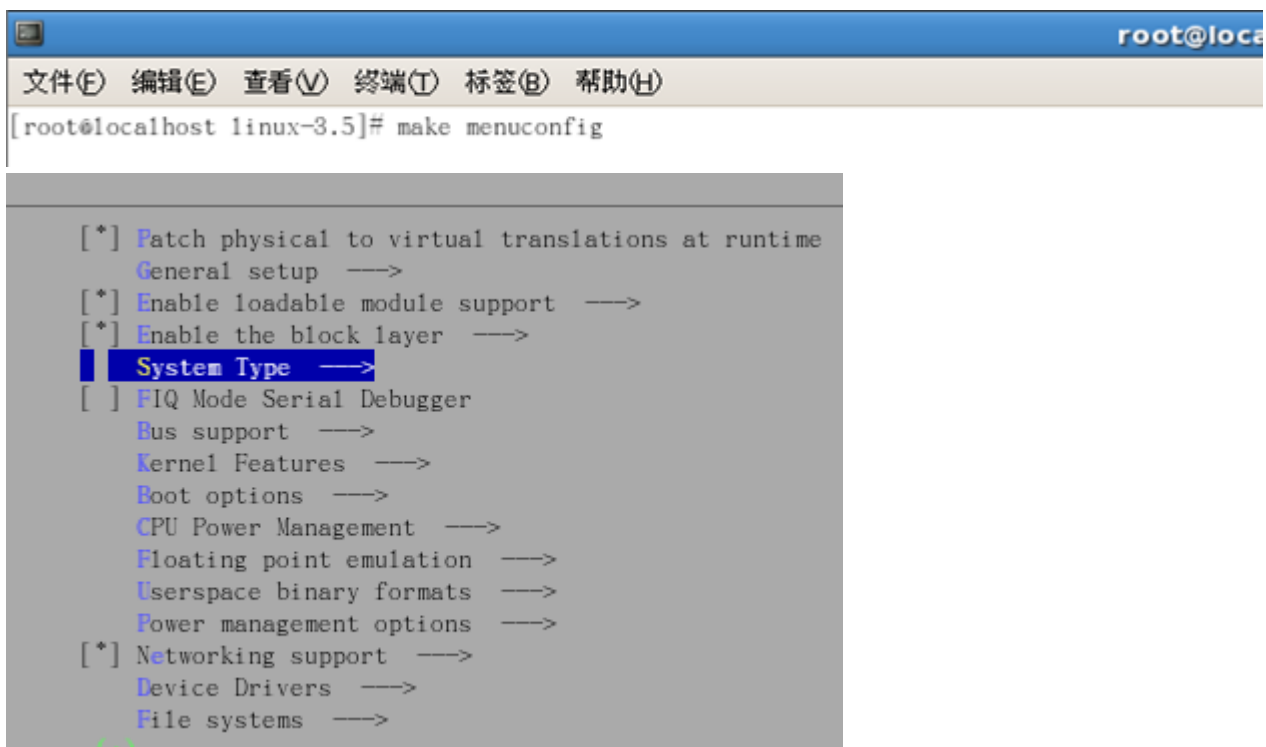
```
[root@localhost ~]# cd linux-3.5
[root@localhost linux-3.5]# ls
arch          fs            MAINTAINERS  security
block         include      Makefile     sound
COPYING       init         mm           tiny4412_android_defconfig
CREDITS       ipc          net          tiny4412_linux_defconfig
crypto        Kbuild       README       tiny4412_ubuntu_defconfig
Documentation Kconfig     REPORTING-BUGS tools
drivers       kernel       samples      usr
firmware      lib          scripts      virt
..
```

3、修改 Makefile 文件 197 行: (指定交叉编译工具)

```
197 #CROSS_COMPILE ?= $(CONFIG_CROSS_COMPILE:"%"=%) (加 #, 屏蔽)
198 CROSS_COMPILE ?= arm-linux-
```

4、执行 `#cp tiny4412_linux_defconfig .config` 配置内核文件，里面全是配置菜单需要的宏；

5、终端字体调小，执行 `#make menuconfig` (字体一定要调小) 读取当前目录下的Kconfig文件，打印图形化菜单，Kconfig读取.config文件里面的宏。



方向键选择第 4 个敲击回车。选择下面图中的选项敲空格。即去除内核编译选项。然后退出保存。

System Type -->

[] Support TrustZone-enabled Trusted Execution Environment

```

[*]
(0) Number of additional GPIO pins
(0) Space between gpio banks
--*-- ADC common driver support
[*] PWM device support
    *** Power management ***
[ ] S3C2410 PM Suspend debug
[ ] S3C2410 PM Suspend Memory CRC
    SAMSUNG EXYNOS SoCs Support -->
    *** Processor Type ***
    *** Processor Features ***
[*] Support TrustZone-enabled Trusted Execution Environment
[ ] Support for the Large Physical Address Extension
[*] Support Thumb user binaries
[*] Enable ThumbEE CPU extension
[*] Emulate SWP/SWPB instructions
[ ] Disable I-Cache (I-bit)

```

6、执行 `#make && make zImage`，等待在 `arch/arm/boot` 里面生成 `zImage`，时间较长(约一小时)；

```
[root@localhost linux-3.5]# make && make zImage
```

```

CALL      scripts/checksyscalls.sh
CHK       include/generated/compile.h
Kernel:  arch/arm/boot/Image is ready
Kernel:  arch/arm/boot/zImage is ready
[root@localhost linux-3.5]# q

```

7、将 `fush_uimage` 文件拷到 `arch/arm/boot` 目录里面，把 `sd` 卡挂接到虚拟机里面。

执行 `./fush_uimage`。最后提示内核烧写成功。

```

[root@localhost boot]# cd /root/linux-3.5/arch/arm/boot/
[root@localhost boot]# pwd
/root/linux-3.5/arch/arm/boot
[root@localhost boot]# ls
bootp  compressed  dts  fush_uimage  Image  install.sh  Makefile  zImage
[root@localhost boot]# ./fush_uimage

```

四、网络文件系统的制作 作用：执行用户程序、提供交互界面；

1、根文件系统目录结构介绍

linux 根文件系统的布局遵循 FHS (Filesystem Hierarchy Standard 文件系统目录标准)，该标准规定了根目录下各个子目录的名称及其存放的内容。

目录名	存放的内容
/bin	必备的用户命令，例如 ls、cp 等
/sbin	必备的系统管理员命令，例如 ifconfig、reboot 等
/dev	设备文件，例如 mtdblock0、tty1 等
/etc	系统配置文件，包括启动文件，例如 inittab 等
/lib	必要的链接库，例如 C 链接库、内核模块
/home	普通用户主目录
/root	root 用户主目录
/usr/bin	非必备的用户程序，例如 find、du 等
/usr/sbin	非必备的管理员程序，例如 chroot、inetd 等
/usr/lib	库文件
/var	守护程序和工具程序所存放的可变，例如日志文件
/proc	用来提供内核与进程信息的虚拟文件系统，由内核自动生成目录下的内容
/sys	用来提供内核与设备信息的虚拟文件系统，由内核自动生成目录下的内容
/mnt	文件系统挂接点，用于临时安装文件系统
/tmp	临时性的文件，重启后将自动清除

2、 根文件系统/etc/目录下重要文件作用分析

2.1、 inittab：在启动过程中 bootloader 会传递参数 init=/linuxrc 给内核的 main()函数，所以在文件系统被挂载后，运行的第一个程序是 linuxrc，而 linuxrc 是一个指向/bin/busybox 的链接文件，也就是说文件系统被挂后运行的第一个程序是 busybox。Busybox 首先会解析文件/etc/inittab，这个文件中存放的是系统的配置信息，这些配置信息指明了接下来将要启动那些程序。

2.2、 init.d/rcS：当解析完文件 etc/inittab 后就将启动这些进程，首先要执行的是启动脚本 etc/init.d/rcS

2.3、 profile：shell 启动时，运行的环境变量设置；

2.4、 fstab：这个文件描述系统中各种文件系统的信息。一般而言，应用程序仅读取这个文件，而不对它进行写操作。对它的维护是系统管理员的工作。在这个文件中，每个文件系统用一行来描述，在每一行中，用空格或 TAB 符号来分隔各个字段，文件中以#开头的行是注释信息。Fstab 文件中

的纪录的排序十分重要。因为 fsck, mount 或 umount 等程序在做它们的工作时会按此顺序进行;

3、 busybox 介绍

BusyBox 是一个集成了一百多个最常用 linux 命令和工具的软件。BusyBox 包含了一些简单的工具,例如 ls、cat 和 echo 等等,还包含了一些更大、更复杂的工具,例 grep、find、mount 以及 telnet。有些人将 BusyBox 称为 Linux 工具里的瑞士军刀。简单的说 BusyBox 就好像是个大工具箱,它集成压缩了 Linux 的许多工具和命令,也包含了 Android 系统的自带的 shell。BusyBox 将许多具有共性的小版本的 UNIX 工具结合到一个单一的可执行文件。这样的集合可以替代大部分常用工具比如的 GNU fileutils , shellutils 等工具, BusyBox 提供了一个比较完善的环境,可以适用于任何小的嵌入式系统。

4、 busybox 制作根文件系统

开发环境: Red Hat Enterprise Linux 5

交叉编译工具: arm-linux-gcc-4.5.1

开发板内核版本: linux-3.5

busybox 软件包版本: busybox-1.21.1.tar.bz2

4.1、将 busybox-1.21.1.tar.bz2 拷贝到虚拟机解压并跳到解压好的目录。

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# tar -xvf busybox-1.21.1.tar.bz2
```

```
[root@localhost ~]# cd busybox-1.21.1
[root@localhost busybox-1.21.1]# ls
applets      console-tools  findutils     loginutils    modutils      selinux
applets_sh   coreutils      include       mailutils     networking    shell
arch          debianutils    init          Makefile      printutils    syslogd
archival     docs           INSTALL       Makefile.custom  procps        testsuite
AUTHORS      e2fsprogs      libbb         Makefile.flags  README        TODO
Config.in    editors        libpwdgrp     Makefile.help   runit         TODO_unicode
configs      examples       LICENSE       miscutils      scripts       util-linux
[root@localhost busybox-1.21.1]#
```

4.2、进入解压的文件夹 busybox-1.21.1 进行安装选项配置,执行 #make menuconfig

4.3、设置共享库

Busybox Settings --->

Build Options --->

[*] Build shared libbusybox

```
Busybox Settings
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

General Configuration --->
Build Options --->
Debugging Options --->
Installation Options ("make install" behavior) --->
Busybox Library Tuning --->

<Select> < Exit > < Help >
```

```
Build Options
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

[ ] Build BusyBox as a static binary (no shared libs) (NEW)
[ ] Build BusyBox as a position independent executable (NEW)
[ ] Force NOMMU build (NEW)
[*] Build shared libbusybox
[*] Produce a binary for each applet, linked against libbusybox (NE
[*] Produce additional busybox binary linked against libbusybox (NE
[*] Build with Large File Support (for accessing files > 2 GB) (NEW)
v(+)

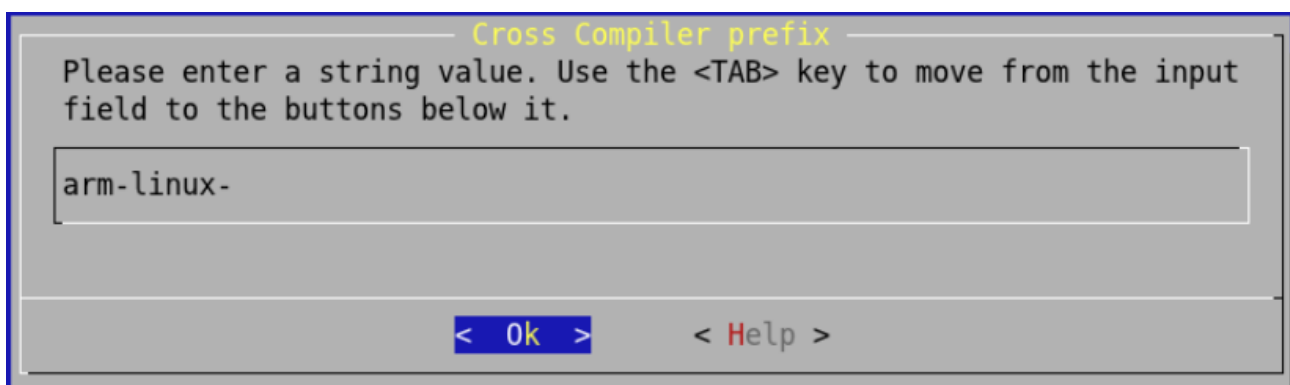
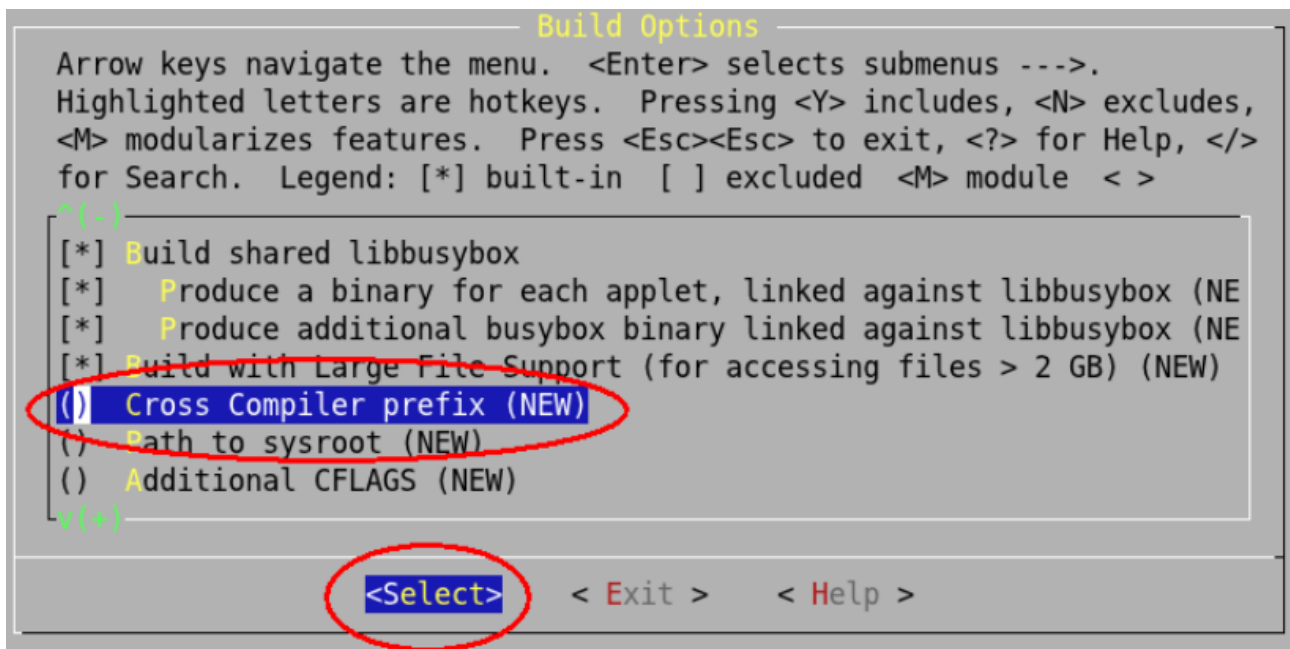
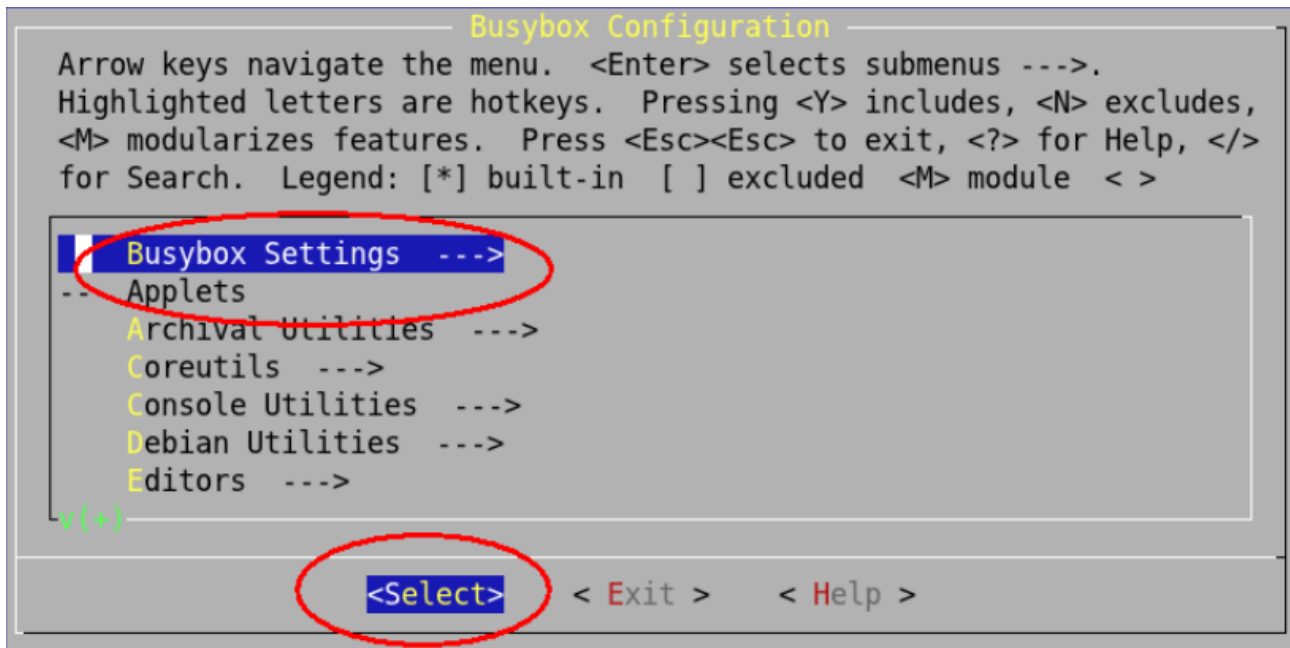
<Select> < Exit > < Help >
```

4.4、指定交叉编译链 (注意 arm-linux-后面不要有空格)

Busybox Settings --->

Build Options --->

() Cross Compiler prefix (NEW)

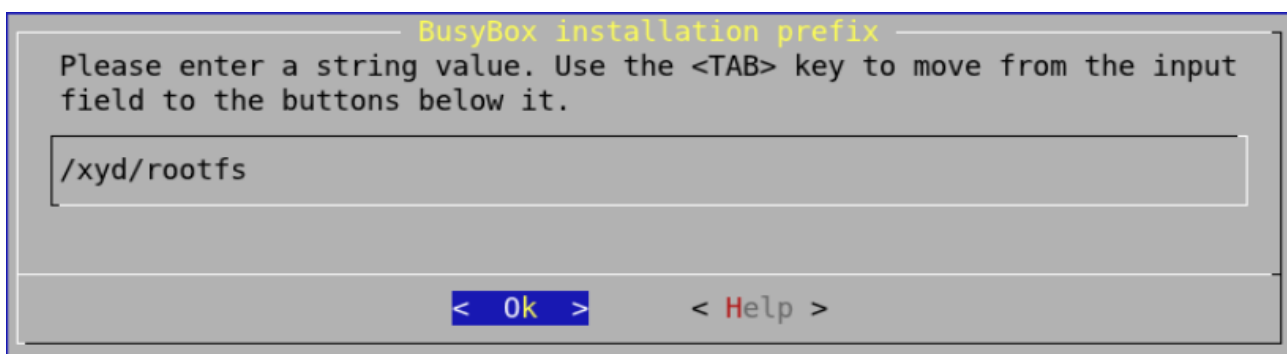
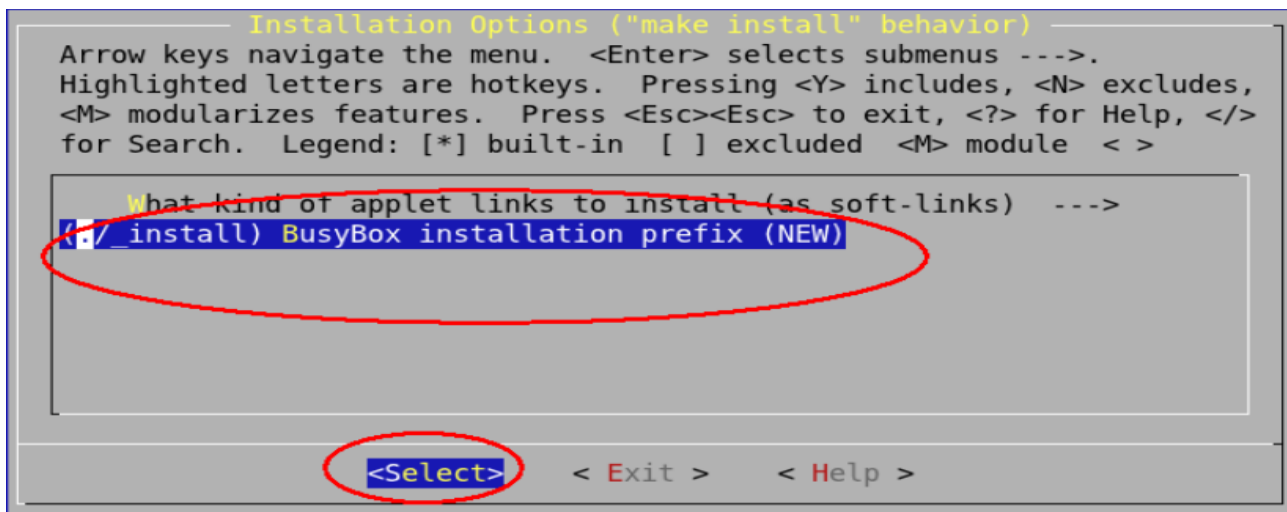


4.5、**设置安装路径**（路径可以自己指定，就是接下来会安装生成的根文件系统，习惯上目录层次不要太深，方便之后使用）

Busybox Settings --->

Installation Options ("make install" behavior) --->

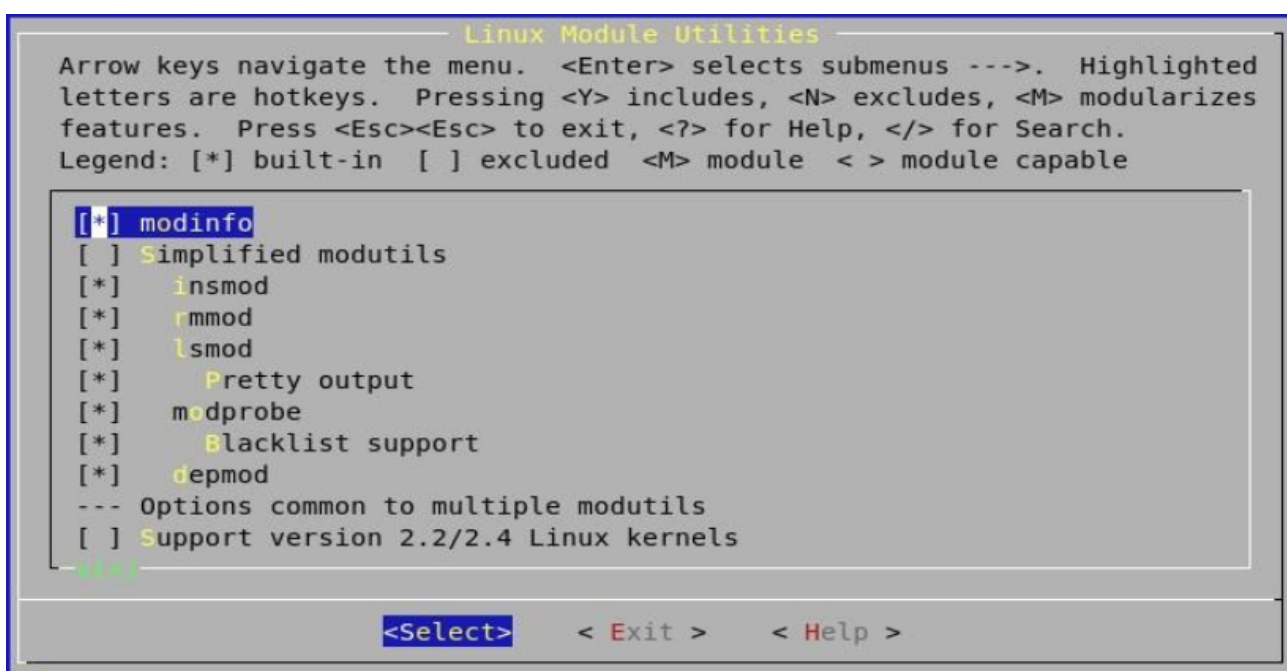
(./install) BusyBox installation prefix(NEW)



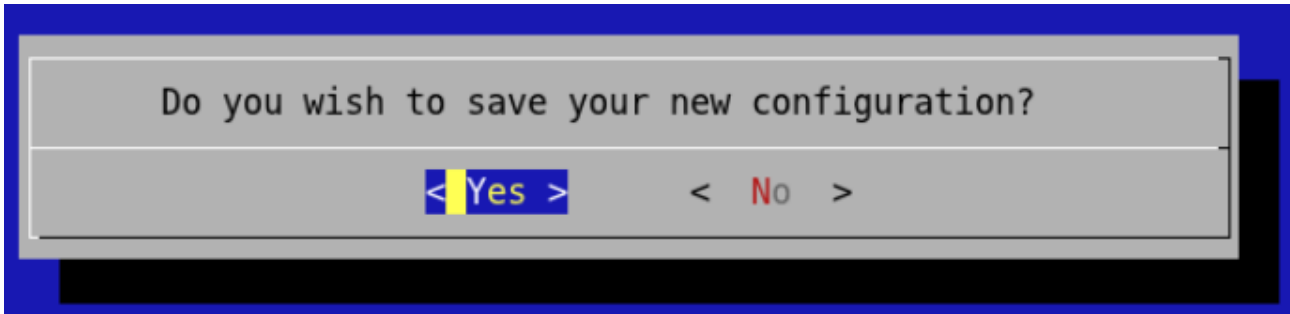
4.6、添加模块指令

#make menuconfig

→ Linux Module Utilities

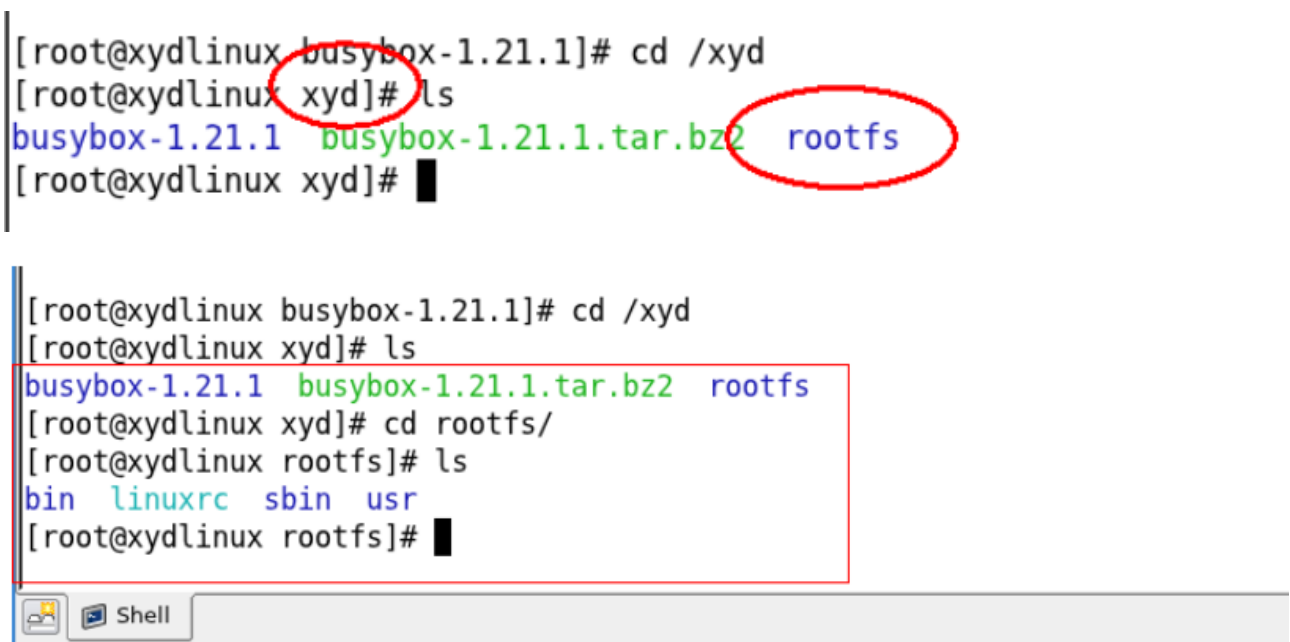


4.7、保存退出。



4.8、编译安装 #make && make install

安装完成之后就会在刚才配置菜单当中指定的路径下找到生成的根文件系统目录 rootfs



4.9、复制命令的动态库： 由于配置 busybox 时候采用动态链接方式编译，所以，要把它所依赖的动态库文件复制到安装目录 rootfs。在根文件系统目录 rootfs 执行：

```
# cp /opt/FriendlyARM/toolschain/4.5.1/arm-none-linux-gnueabi/lib/ ./ -rap
```



4.10、创建其他目录， 生成完整根文件系统目录结构

```
# mkdir dev etc/init.d home proc sys root opt tmp var -p
```

4.11、在“根文件系统”（这里对应 **rootfs**）创建设备节点

```
# mknod dev/console c 5 1
```

```
# mknod dev/null c 1 3
```

```
[root@xydlinux rootfs]# mknod dev/console c 5 1
[root@xydlinux rootfs]# mknod dev/null c 1 3
[root@xydlinux rootfs]# ls -l dev/
总计 0
crw-r--r-- 1 root root 5, 1 06-29 17:14 console
crw-r--r-- 1 root root 1, 3 06-29 17:14 null
[root@xydlinux rootfs]#
```

4.12、构建 **etc** 目录下的系统配置文件

A、创建 **fstab** 文件 # cp /etc/fstab ./etc

```
[root@xydlinux rootfs]# ls
bin dev etc home lib linuxrc opt proc root sbin sys tmp usr var
[root@xydlinux rootfs]# cp /etc/fstab ./etc
[root@xydlinux rootfs]# vim ./etc/fstab
```

修改 **fstab** 文件内容为以下（注意类型顺序）：

```
# device mount-point type options dump fsck order
proc /proc proc defaults 0 0
tmpfs /tmp tmpfs defaults 0 0
sysfs /sys sysfs defaults 0 0
tmpfs /dev tmpfs defaults 0 0
```

# device	mount-point	type	options	dump	fsck	order
proc	/proc	proc	defaults	0	0	
tmpfs	/tmp	tmpfs	defaults	0	0	
sysfs	/sys	sysfs	defaults	0	0	
tmpfs	/dev	tmpfs	defaults	0	0	

B、创建 **inittab** 文件

```
# cp /xyd/busybox-1.21.1/examples/inittab ./etc/
```

```
# vim etc/inittab
```

```
[root@xydlinux rootfs]# cp /xyd/busybox-1.21.1/examples/inittab ./etc/
[root@xydlinux rootfs]# vim etc/inittab
```

修改 `inittab` 文件内容为以下:

```
::sysinit:/etc/init.d/rcS 启动系统初始化文件/etc/init.d/rcS
console::askfirst:/bin/sh 在串口终端启动 askfirst 动作的 shell
::ctrlaltdel:/sbin/reboot 当按下 Ctrl+Alt+Delete 组合键时, init 重启执行程序
::shutdown:/bin/umount -a -r 关机时运行 umount 命令卸载所有的文件系统, 如果卸载失败, 试图以只读方式重新挂载
```



```
root@xydlinux:/xyd/rootfs/etc - Shell - Konsole
会话 编辑 查看 书签 设置 帮助

::sysinit:/etc/init.d/rcS
console::askfirst:/bin/sh
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r
```

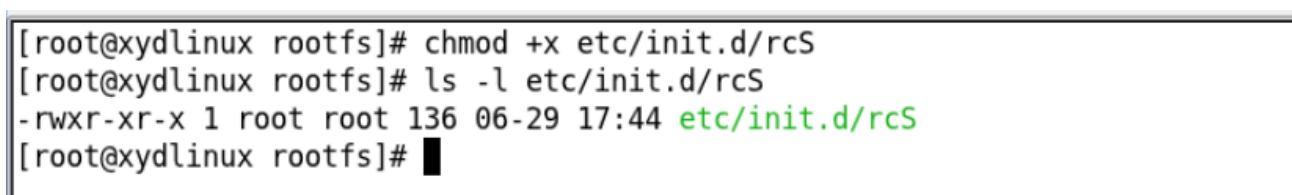
C、创建 `etc/init.d/rcS` 文件, 添加内容如下:



```
[root@xydlinux rootfs]# touch etc/init.d/rcS
[root@xydlinux rootfs]# vim etc/init.d/rcS

#!/bin/sh 声明 shell 脚本类型, 使用 busybox 的 shell
mount -a 将文件 /etc/fstab 中指定的文件挂载到对应的挂载点
mkdir /dev/pts
mount -t devpts devpts /dev/pts
echo /sbin/mdev > /proc/sys/kernel/hotplug 热插拔事件时产生设备节点的支持
mdev -s
/bin/hostname XYD
```

并且给 `rcS` 文件添加执行权限 `# chmod +x etc/init.d/rcS`



```
[root@xydlinux rootfs]# chmod +x etc/init.d/rcS
[root@xydlinux rootfs]# ls -l etc/init.d/rcS
-rwxr-xr-x 1 root root 136 06-29 17:44 etc/init.d/rcS
[root@xydlinux rootfs]#
```

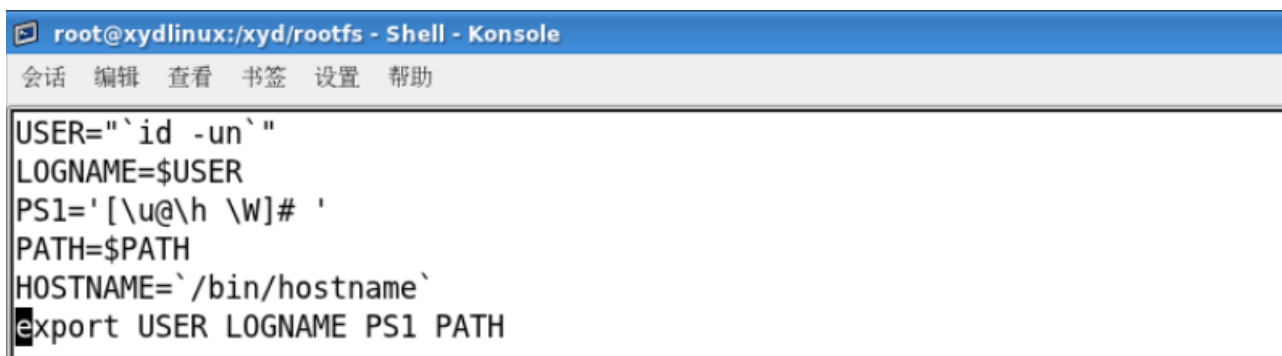
D、创建 `etc/profile` 文件



```
[root@xydlinux rootfs]# touch etc/profile
[root@xydlinux rootfs]# vim etc/profile
```

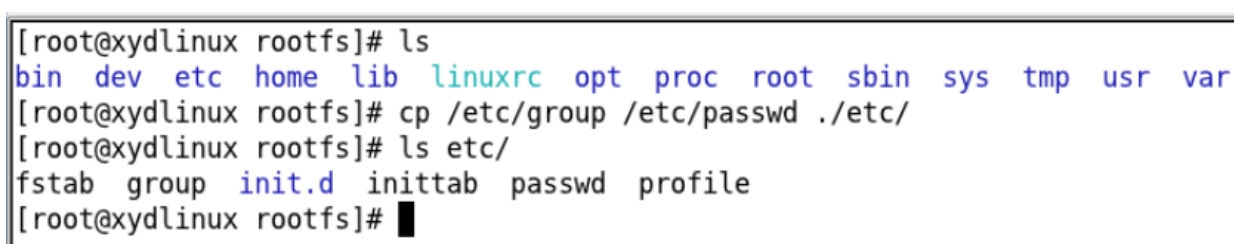
添加内容为：

```
USER="`id -un`"  
LOGNAME=$USER  
PS1='[\u@\h \W]# '  
PATH=$PATH  
HOSTNAME=`/bin/hostname`  
export USER LOGNAME PS1 PATH
```



实现提示符功能还需要两个文件的支持,拷贝虚拟机 **linux** 目录的 `etc/group` 和 `etc/passwd` 的文

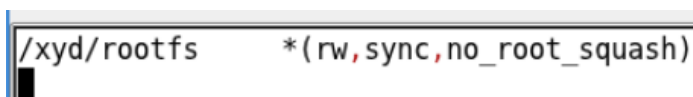
件到根文件系统 `# cp /etc/group /etc/passwd ./etc/`



4.13、编辑`/etc/exports` 文件, `# vim /etc/exports` ；

加入允许其它计算机访问的目录和访问权限 (注意是虚拟机 **linux** 的 `etc` 目录)

文件内容: `/xyd/rootfs *(rw, sync, no_root_squash)`



注意：

目录与 IP 之间用<Tab>键

说明：

<code>/xyd/rootfs</code>	允许其他计算机访问的目录
<code>*</code>	被允许访问该目录的客户端的 IP 地址（如果限制 IP 访问，可以自己添加限制 IP，例如： <code>/xyd/rootfs 192.168.15.*(rw, sync, no_root_squash)</code> ）
<code>rw</code>	可读写权限
<code>sync</code>	同步写磁盘（ <code>async</code> ：资料会先暂存于内存当中，而非直接写入硬盘）
<code>no_root_squash</code>	表示客户端 <code>root</code> 用户对该目录具备写权限

4.14、启动 NFS 服务 # /etc/init.d/nfs start 或者 # service nfs start

重启 NFS 服务命令类似 (service nfs restart 或 /etc/init.d/nfs restart)

4.14、设置 NFS 开机自动启动 # chkconfig nfs on

4.15、开发板 Linux 内核挂载 NFS 文件系统相关环境配置的操作步骤与出错排查

(如果挂载不上,排除根文件系统制作错误之外, 检查以下设置↓↓↓)

A、设置虚拟机网络 IP (注意虚拟机、物理机、开发板 linux 内核的 IP 为同一号段)

service network restart

ifconfig

这里用的是 192.168.15.2

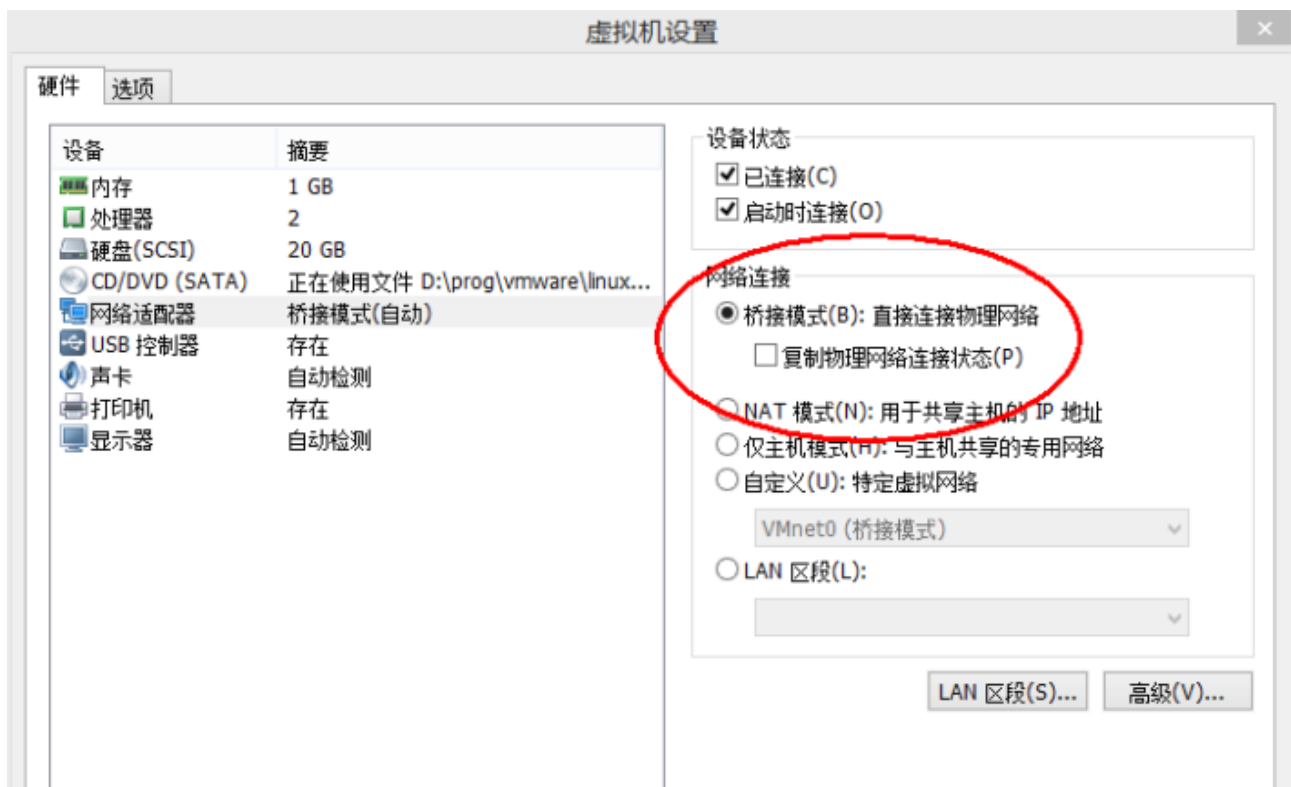
```
[root@xydlinux rootfs]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:33:C7:BA
          inet addr:192.168.15.2  Bcast:192.168.15.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe33:c7ba/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:52120 errors:0 dropped:0 overruns:0 frame:0
          TX packets:102004 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9224571 (8.7 MiB)  TX bytes:127069352 (121.1 MiB)
          Interrupt:67 Base address:0x2024

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:10653 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10653 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4139824 (3.9 MiB)  TX bytes:4139824 (3.9 MiB)

[root@xydlinux rootfs]# █
```

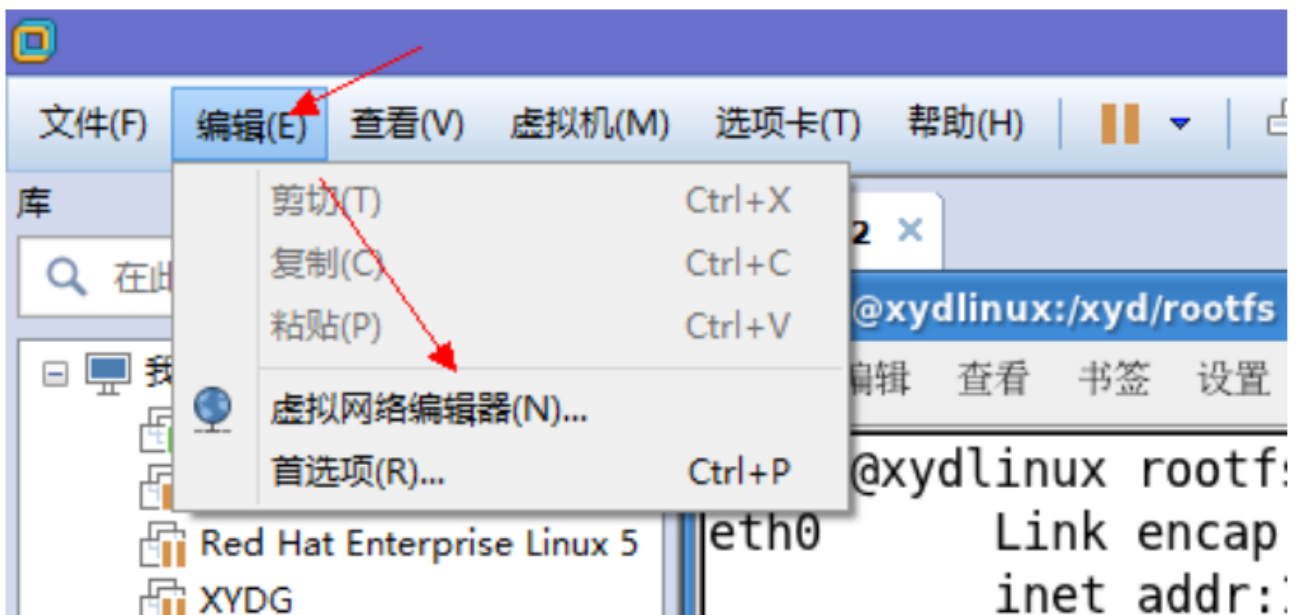
B、虚拟机网络设置,配置为“桥接”:

VMware 右下角网络适配器—选择“设置”—更改为“桥接模式”



C、虚拟机 VMware 选项卡“编辑”--“虚拟网络编辑器”--桥接设置桥接到物理网络

注意：如果是笔记本电脑，这里桥接到物理机的物理网卡，不要桥接到无线网卡，各个电脑网卡型号有所不同，看自己情况选择





D、重启网络 # service network restart

E、关闭防火墙 # service iptables stop

F、重启 NFS 服务 # service nfs restart

G、挂接不成功时记得查看根文件系统 etc 目录下的 rcS、fstab、inittab 等文件是否有执行权限，如果没有，加执行权限

H、配置物理机网络连接：

windows 打开网络和共享中心—更改适配器设置--以太网/本地连接--右击属性--IPv4，将 IP 设置为虚拟机同一号段，这里是 192.168.15.16

I、关闭 windows 防火墙：（在控制面板也可以找到）

网络共享中心 — Windows 防火墙 — 启用或关闭 windows 防火墙 --关闭 — 确定

J、进入 uboot 修改 bootargs ，uboot--kernel--根文件系统

```
#set bootargs 'noinitrd root=/dev/nfs nfsroot=192.168.15.2:/xyd/rootfs
ip=192.168.15.5:192.168.15.2:192.168.15.1:255.255.255.0::eth0:off init=/linuxrc
console=ttySAC0 lcd=S70'
```

save

reset

```
xyd4412 >
xyd4412 >
xyd4412 >
xyd4412 >
xyd4412 >
xyd4412 >
xyd4412 > print
baudrate=115200
bootargs=noinitrd root=/dev/nfs nfsroot=192.168.15.2:/rec/fs_root/busybox-1.21.1/_install ip=192.168.15.5:192.168.15.2:192.168.15.1:255.255.255.0::eth0:off init=/linuxrc console=ttySAC0 lcd=S70
bootcmd=movi read kernel 0 40007fc0;bootm 40007fc0
bootdelay=3
ethaddr=00:40:5c:26:0a:5b
gatewayip=192.168.0.1
ipaddr=192.168.0.20
netmask=255.255.255.0
serverip=192.168.0.10

Environment size: 404/16380 bytes
xyd4412 > #set bootargs 'noinitrd root=/dev/nfs nfsroot=192.168.15.2:/xyd/rootfs ip=192.168.15.5:192.168.15.2:192.168.15.1:255.255.255.0::eth0:off init=/linuxrc console=ttySAC0 lcd=S70'
xyd4412 > save
Saving Environment to SMDK bootable device...
done
xyd4412 > reset
```

END: 挂载成功提示

[10.090000] VFS: Mounted root (nfs filesystem) on device 0:10.

[10.090000] Freeing init memory: 212K

[11.855000] nf_conntrack: automatic helper assignment is deprecated and it will be removed soon. Use the iptables CT target to attach helpers instead.

Please press Enter to activate this console. 按回车进入根文件系统

```
[ 5.295000] link_reset() speed: 10 duplex: 0
[ 5.300000] IP-Config: Complete:
[ 5.300000] device=eth0, addr=192.168.15.5, mask=255.255.255.0, gw=192.168.15.1
[ 5.300000] host=192.168.15.5, domain=, nis-domain=(none)
[ 5.300000] bootserver=192.168.15.2, rootserver=192.168.15.2, rootpath=
[ 5.305000] hotplug_policy_init: initialised with policy : DVFS_NR_BASED_HOTPLUG
[ 5.310000] ALSA device list:
[ 5.315000] No soundcards found.
[ 6.440000] VFS: Mounted root (nfs filesystem) on device 0:11.
[ 6.440000] devtmpfs: mounted
[ 6.440000] Freeing init memory: 216K
[ 8.675000] nf_conntrack: automatic helper assignment is deprecated and it will be removed soon. Use the iptables CT target to attach helpers instead.

Please press Enter to activate this console.
[root@XYD /]#
[root@XYD /]#
[root@XYD /]#
[root@XYD /]#
[root@XYD /]#
[root@XYD /]#
[root@XYD /]# ls
bin  etc  lib  opt  root  sys  usr
dev  home linuxrc proc sbin tmp  var
[root@XYD /]#
```

- 1、制作启动盘;
- 2、烧写 uboot 找到 sdb `sd_fuse/sd_fuse.sh ./sd_fuse.sh /dev/sdb` ;
- 3、烧写内核 `linux-3.5/arch/arm/boot ./fuse_image` ;
- 4、创建网络文件系统;

做到这一步的时候还没出问题，恭喜你已基本完成了，剩下的就是你挂接的问题了！

- 1、启动 NFS 并开机自启动

```
# service nfs start
```

```
#chkconfig nfs on
```

- 2、设置网络：你要设置的网络有虚拟机的网络、物理机的网络、uboot 传递的参数

虚拟机网络：1、选择桥接；2、配置静态 IP : 192.168.0.101

配置完毕后重启下网络使其生效#service network restart

物理机网络的本地连接的 IPv4 配置为：192.168.0.100

配置 uboot 传递参数，uboot 启动的时候敲回车进入：

```
#set          bootargs          'noinitrd          root=/dev/nfs          nfsroot=192.168.0.101:/xyd/rootnfs
```

```
ip=192.168.0.101:192.168.0.100:192.168.0.1:255.255.255.0::eth0:off init=/linuxrc console=ttySAC0 lcd=S70'
```

```
#save
```

- 3、插上网线上电就可以了，所有工作完毕。