

中国地质大学

本科生实验报告



课程名称	<u>嵌入式与可编程技术 II</u>
教师姓名	<u>张莉君 刘玮</u>
学生姓名	<u>陈泓旭</u>
学生学号	<u>20191003795</u>
所在班级	<u>231201</u>
所在专业	<u>自动化</u>
日期	<u>2022 年 10 月 22 日</u>

目 录

第二篇 Linux 环境下裸机开发实验.....	1
一、实验目的	1
二、实验要求	1
三、实验过程	1
3.1 实验五 熟悉 linux 操作系统	1
3.2 实验六 linux 常用命令的熟悉	4
3.3 实验八 编译工具 GCC 的使用	6
3.4 实验九 交叉编译工具的使用	9
3.5 实验十 linux 环境下裸机程序设计及编译	12
第三篇 Linux 操作系统移植与驱动初体验	14
一、实验目的	14
二、实验要求	14
三、实验内容	14
3.1 实验十一 Linux 系统移植初体验	14
3.2 实验十一 Linux 操作系统下驱动开发初体验	16
四、思考题	17
五、心得与体会	18

第二篇 Linux 环境下裸机开发实验

一、实验目的

熟悉 Linux 系统环境，掌握常用命令，学会使用 Linux 环境下开发工具，了解应用程序的开发流程。

二、实验要求

- 1、结合实验五，说说你所理解的 Linux 操作系统；
- 2、列出你在实验六中执行的一些常规命令行，给出该行命令的功能解释，并附上命令行执行截图；
- 3、列出实验六、实验八的实验过程，编写并编译自己的第一个 c 程序（硬件无关），并附实验截图；
- 4、列出实验九、十的实验过程，并附实验截图；

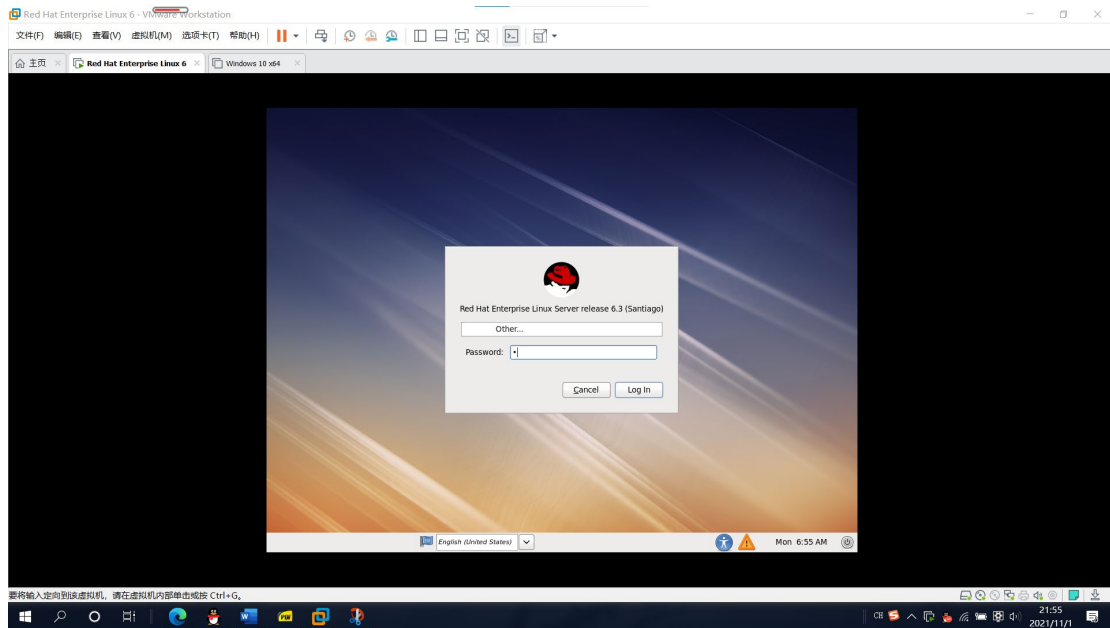
三、实验过程

3.1 实验五 熟悉 linux 操作系统

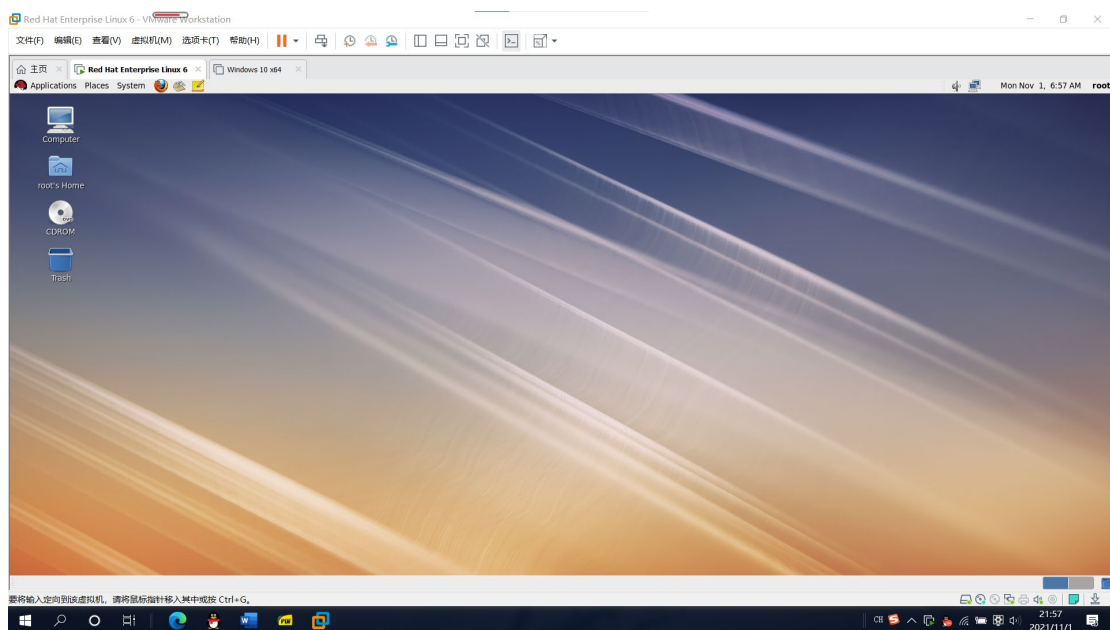
1) 打开虚拟机



2) 登录 Linux 操作系统

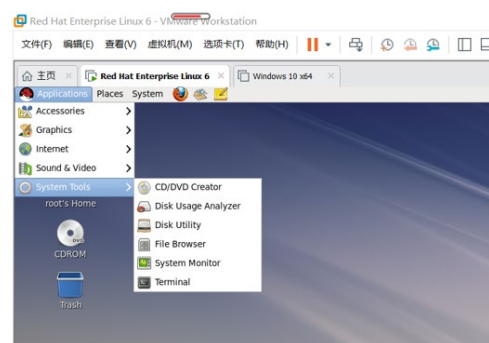


进入 Linux 系统桌面环境如下图所示：

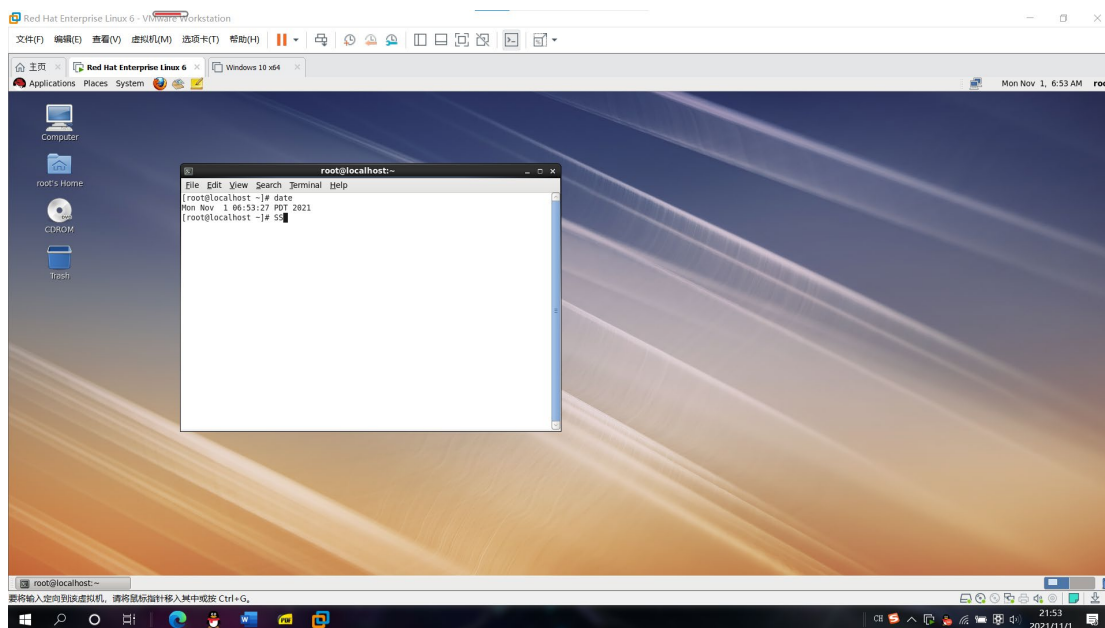


3) 打开命令终端 (Terminal)

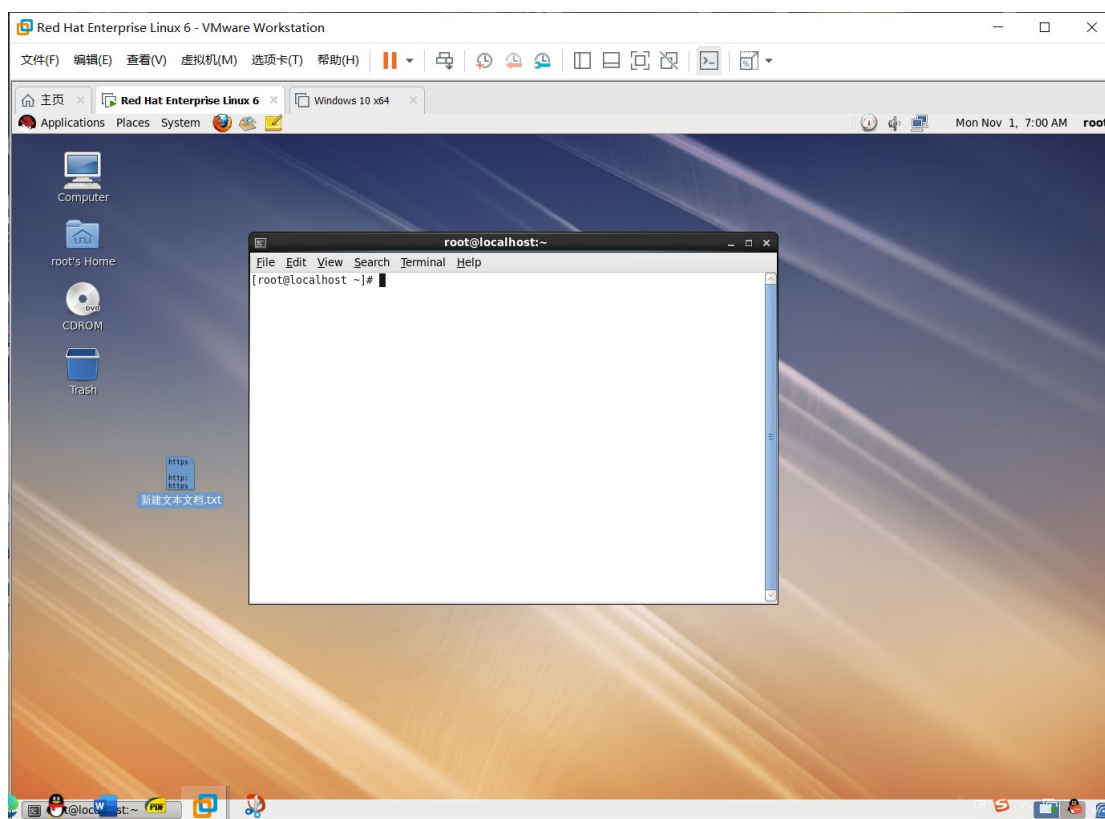
单击 Applications->System Tools->Terminal，打开命令行终端，如下图所示：



在 Linux 终端中写入第一个命令 `date`，回车后，终端显示出当前的日期和时间如下图所示。



文件共享：



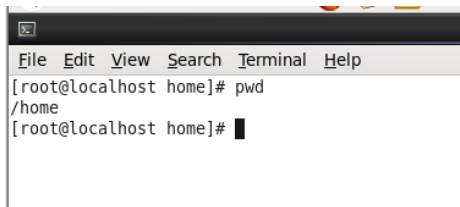
结合实验五，我对 Linux 的理解：

Linux 和 Windows 有相似之处，比如都有用户交互界面，有菜单按钮等等；不过，Linux 调用系统的功能是可以依靠命令行进行的。linux 是一个命令行组成的操作系统，我们可以通过对命令行写入一句句命令来执行每一步操作，我们需要学习如何执行系统命令，包括有关文件、目录、文件系统、进程等概念，如何使用相应的命令对文件、目录、进程等进行管

理，了解遇到问题时，如何找到帮助信息等。

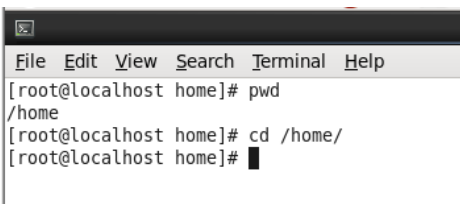
3.2 实验六 linux 常用命令的熟悉

1) 使用 pwd 显示工作目录:

A terminal window with a menu bar (File, Edit, View, Search, Terminal, Help) and a title bar. The prompt is [root@localhost home]#. The user enters 'pwd' and the output is '/home'.

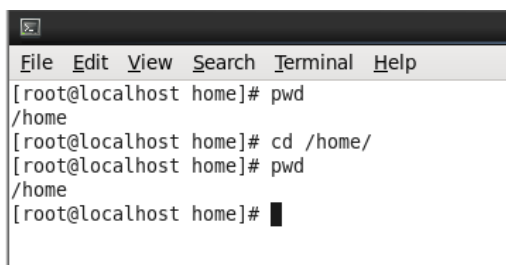
```
[root@localhost home]# pwd
/home
[root@localhost home]#
```

2) 使用 cd 切换到 home 目录:

A terminal window with a menu bar (File, Edit, View, Search, Terminal, Help) and a title bar. The prompt is [root@localhost home]#. The user enters 'pwd' and the output is '/home'. Then the user enters 'cd /home/' and the prompt changes to [root@localhost home]#.

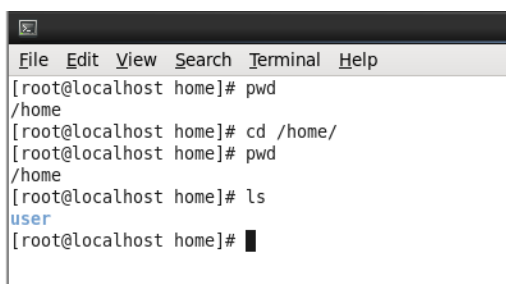
```
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]#
```

3) 使用 cd 确认当前是否在 home 目录:

A terminal window with a menu bar (File, Edit, View, Search, Terminal, Help) and a title bar. The prompt is [root@localhost home]#. The user enters 'pwd' and the output is '/home'. Then the user enters 'cd /home/' and the prompt changes to [root@localhost home]#. Finally, the user enters 'pwd' and the output is '/home'.

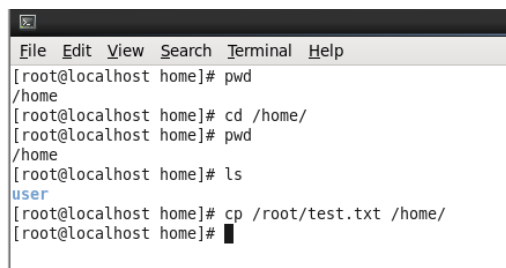
```
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]# pwd
/home
[root@localhost home]#
```

4) 使用 ls 列出 home 目录中的内容:

A terminal window with a menu bar (File, Edit, View, Search, Terminal, Help) and a title bar. The prompt is [root@localhost home]#. The user enters 'pwd' and the output is '/home'. Then the user enters 'cd /home/' and the prompt changes to [root@localhost home]#. Finally, the user enters 'ls' and the output is 'user'.

```
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]# pwd
/home
[root@localhost home]# ls
user
[root@localhost home]#
```

5) 使用 cp 指令，把 root 目录下的 test.txt 复制到 home 目录下:

A terminal window with a menu bar (File, Edit, View, Search, Terminal, Help) and a title bar. The prompt is [root@localhost home]#. The user enters 'pwd' and the output is '/home'. Then the user enters 'cd /home/' and the prompt changes to [root@localhost home]#. Finally, the user enters 'cp /root/test.txt /home/' and the prompt changes to [root@localhost home]#.

```
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]# pwd
/home
[root@localhost home]# ls
user
[root@localhost home]# cp /root/test.txt /home/
[root@localhost home]#
```

6) 使用 ls 确认复制是否成功:

```
File Edit View Search Terminal Help
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]# pwd
/home
[root@localhost home]# ls
user
[root@localhost home]# cp /root/test.txt /home/
[root@localhost home]# ls
test.txt user
[root@localhost home]#
```

7) 使用 mv 指令将 home 目录下的 test.txt 重命名为 newname.txt:

```
File Edit View Search Terminal Help
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]# pwd
/home
[root@localhost home]# ls
user
[root@localhost home]# cp /root/test.txt /home/
[root@localhost home]# ls
test.txt user
[root@localhost home]# mv test.txt newname.txt
[root@localhost home]#
```

8) 使用 ls 确认重命名是否成功

```
File Edit View Search Terminal Help
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]# pwd
/home
[root@localhost home]# ls
user
[root@localhost home]# cp /root/test.txt /home/
[root@localhost home]# ls
test.txt user
[root@localhost home]# mv test.txt newname.txt
[root@localhost home]# ls
newname.txt user
[root@localhost home]#
```

9) 使用 rm 删除刚刚重命名的文件:

```
File Edit View Search Terminal Help
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]# pwd
/home
[root@localhost home]# ls
user
[root@localhost home]# cp /root/test.txt /home/
[root@localhost home]# ls
test.txt user
[root@localhost home]# mv test.txt newname.txt
[root@localhost home]# ls
newname.txt user
[root@localhost home]# rm newname.txt
rm: remove regular file 'newname.txt'? y
You have new mail in /var/spool/mail/root
[root@localhost home]#
```

10) 使用 ls 确认文件是否删除成功:

```
File Edit View Search Terminal Help
[root@localhost home]# pwd
/home
[root@localhost home]# cd /home/
[root@localhost home]# pwd
/home
[root@localhost home]# ls
user
[root@localhost home]# cp /root/test.txt /home/
[root@localhost home]# ls
test.txt user
[root@localhost home]# mv test.txt newname.txt
[root@localhost home]# ls
newname.txt user
[root@localhost home]# rm newname.txt
rm: remove regular file `newname.txt'? y
You have new mail in /var/spool/mail/root
[root@localhost home]# ls
user
[root@localhost home]#
```

3.3 实验八 编译工具 GCC 的使用

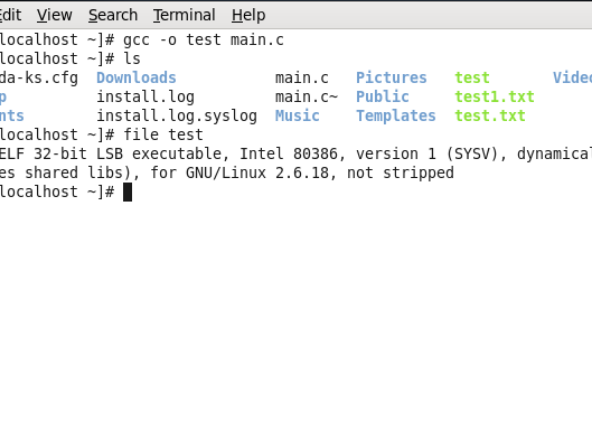
1) gcc 编译代码和执行:

①使用 gcc 利用 main.c 生成 test 文件:

```
root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# gcc -o test main.c
[root@localhost ~]#
```

```
root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# gcc -o test main.c
[root@localhost ~]# ls
anaconda-ks.cfg  Downloads      main.c  Pictures  test      Videos
Desktop          install.log    main.c~ Public    test1.txt
Documents        install.log.syslog Music    Templates test.txt
[root@localhost ~]#
```

②查看 test 文件信息:



The screenshot shows a terminal window titled "root@localhost:~". The terminal content is as follows:

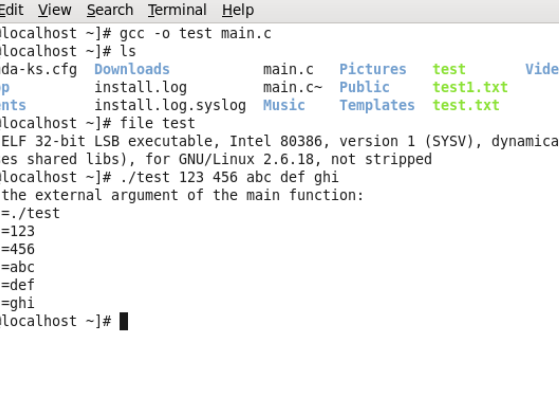
```

File Edit View Search Terminal Help
[root@localhost ~]# gcc -o test main.c
[root@localhost ~]# ls
anaconda-ks.cfg  Downloads          main.c      Pictures      test         Videos
Desktop          install.log        main.c~     Public        test1.txt
Documents        install.log.syslog Music         Templates    test.txt
[root@localhost ~]# file test
test: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link
ed (uses shared libs), for GNU/Linux 2.6.18, not stripped
[root@localhost ~]#

```

The terminal output shows the successful compilation of the C program into an executable named "test". The "ls" command lists the files in the current directory, including the source file "main.c", the compiled executable "test", and various system files. The "file" command confirms that "test" is a 32-bit ELF executable for Intel 80386 architecture, dynamically linked, and not stripped.

③运行可执行文件:



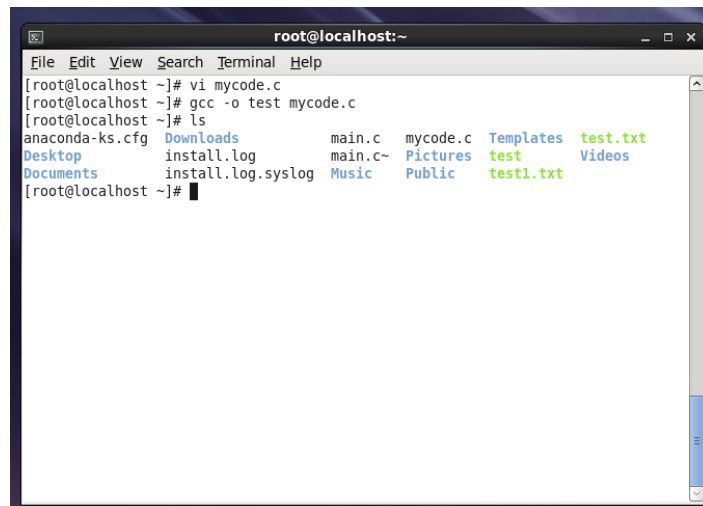
```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# gcc -o test main.c  
[root@localhost ~]# ls  
anaconda-ks.cfg  Downloads          main.c  Pictures  test      Videos  
Desktop          install.log       main.c~ Public    test1.txt  
Documents        install.log.syslog Music     Templates test.txt  
[root@localhost ~]# file test  
test: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link  
ed (uses shared libs), for GNU/Linux 2.6.18, not stripped  
[root@localhost ~]# ./test 123 456 abc def ghi  
Print the external argument of the main function:  
arg[1]=./test  
arg[2]=123  
arg[3]=456  
arg[4]=abc  
arg[5]=def  
arg[6]=ghi  
[root@localhost ~]#
```

2) 编写自己的 C 文件, 并使用 gcc 编译执行:

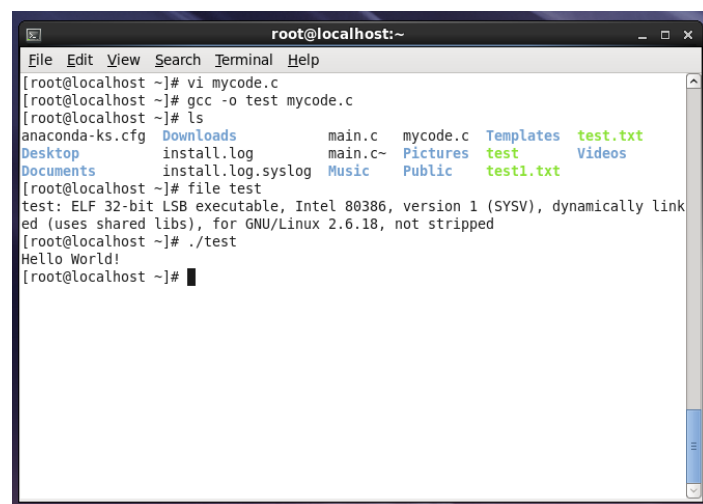
①编写文件:

[illegible]

②使用 gcc 编译执行:

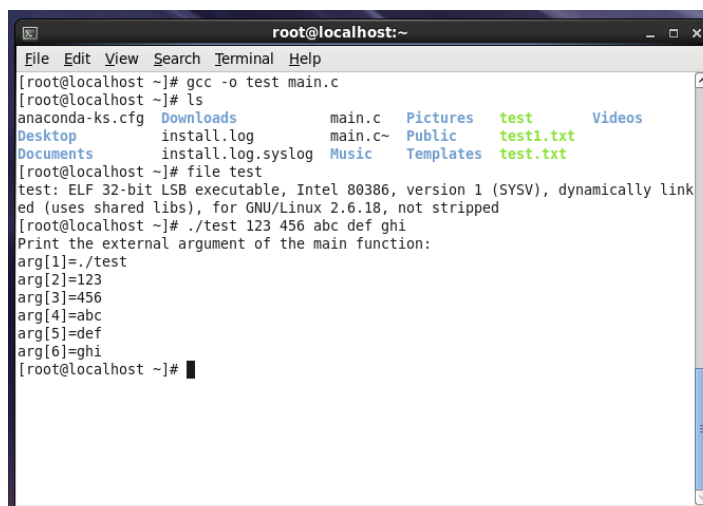


```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# vi mycode.c  
[root@localhost ~]# gcc -o test mycode.c  
[root@localhost ~]# ls  
anaconda-ks.cfg Downloads main.c mycode.c Templates test.txt  
Desktop install.log main.c~ Pictures test Videos  
Documents install.log.syslog Music Public test1.txt  
[root@localhost ~]#
```



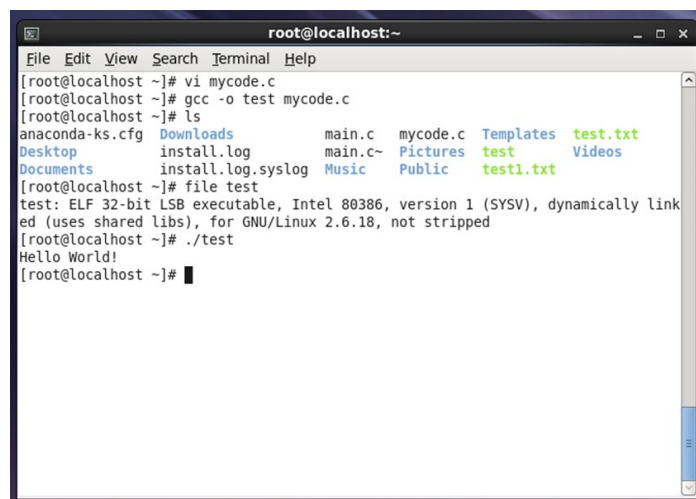
```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# vi mycode.c  
[root@localhost ~]# gcc -o test mycode.c  
[root@localhost ~]# ls  
anaconda-ks.cfg Downloads main.c mycode.c Templates test.txt  
Desktop install.log main.c~ Pictures test Videos  
Documents install.log.syslog Music Public test1.txt  
[root@localhost ~]# file test  
test: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link  
ed (uses shared libs), for GNU/Linux 2.6.18, not stripped  
[root@localhost ~]# ./test  
Hello World!  
[root@localhost ~]#
```

初次使用 gcc 编译代码并执行后的结果如下：



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# gcc -o test main.c  
[root@localhost ~]# ls  
anaconda-ks.cfg Downloads main.c Pictures test Videos  
Desktop install.log main.c~ Public test1.txt  
Documents install.log.syslog Music Templates test.txt  
[root@localhost ~]# file test  
test: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link  
ed (uses shared libs), for GNU/Linux 2.6.18, not stripped  
[root@localhost ~]# ./test 123 456 abc def ghi  
Print the external argument of the main function:  
arg[1]=./test  
arg[2]=123  
arg[3]=456  
arg[4]=abc  
arg[5]=def  
arg[6]=ghi  
[root@localhost ~]#
```

编写自己的 C 文件并编译执行的结果如下：



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# vi mycode.c  
[root@localhost ~]# gcc -o test mycode.c  
[root@localhost ~]# ls  
anaconda-ks.cfg  Downloads          main.c    mycode.c  Templates  test.txt  
Desktop          install.log        main.c~   Pictures  test       Videos  
Documents        install.log.syslog Music      Public    test1.txt  
[root@localhost ~]# file test  
test: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link  
ed (uses shared libs), for GNU/Linux 2.6.18, not stripped  
[root@localhost ~]# ./test  
Hello World!  
[root@localhost ~]#
```

3.4 实验九 交叉编译工具的使用

1) 安装交叉编译工具

输入解压命令：# tar xvzf arm-linux-gcc-4.4.3-20100728.tar.gz -C /后，结果如下图：

```

File Edit View Search Terminal Help
chdog_Library_1_Handler_Flag-members.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/ftv2doc.png
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/ftv2mlastnode.png
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/ftv2lastnode.png
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/ftv2link.png
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_EList_Iterator.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/tree.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/annotated.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/functions.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/GFDL.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/tab_l.gif
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_Watchdog-members.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/ftv2node.png
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_Pending_Element.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_Watchdog.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/pages.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_Handler.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/graph_legend.png
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_Handler_Function-members.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_Doubly_Linked_Object.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_Handler_Flag.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/classParma_Watchdog_Library_1_EList-members.html
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7-html/doxygen.png
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/fdl.ps.gz
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7.ps.gz
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/gpl.txt
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/README
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/pwl-user-0.7.pdf
opt/FriendlyARM/toolschain/4.4.3/share/doc/pwl/gpl.ps.gz
opt/FriendlyARM/toolschain/4.4.3/share/aclocal/
opt/FriendlyARM/toolschain/4.4.3/share/aclocal/ppl_c.m4
opt/FriendlyARM/toolschain/4.4.3/share/aclocal/ppl.m4
[root@localhost ~]#

```

2) 修改系统环境变量:

执行# vi /etc/profile 指令后, 如下图所示:

```
root@localhost:~  
File Edit View Search Terminal Help  
/etc/profile  
  
# System wide environment and startup programs, for login setup  
# Functions and aliases go in /etc/bashrc  
  
# It's NOT a good idea to change this file unless you know what you  
# are doing. It's much better to create a custom.sh shell script in  
# /etc/profile.d/ to make custom changes to your environment, as this  
# will prevent the need for merging in future updates.  
  
pathmunge () {  
    case "${PATH}" in  
        *:"$1":*)  
            ;;  
        *)  
            if [ "$2" = "after" ] ; then  
                PATH=$PATH:$1  
            else  
                PATH=$1:$PATH  
            fi  
        esac  
    }  
  
if [ -x /usr/bin/id ]; then  
    if [ -z "$EUID" ]; then  
        # ksh workaround  
        EUID=`id -u`  
        UID=`id -ru`  
    fi  
    USER=`id -un`  
    LOGNAME=$USER  
    MAIL="/var/spool/mail/$USER"  
fi  
  
# Path manipulation  
if [ "$EUID" = "0" ]; then  
    pathmunge /sbin  
    pathmunge /usr/sbin  
    pathmunge /usr/local/sbin  
else  
    pathmunge /usr/local/sbin after  
    pathmunge /usr/sbin after  
    pathmunge /sbin after
```

```
root@localhost:~  
File Edit View Search Terminal Help  
# Path manipulation  
if [ "$EUID" = "0" ]; then  
    pathmunge /sbin  
    pathmunge /usr/sbin  
    pathmunge /usr/local/sbin  
else  
    pathmunge /usr/local/sbin after  
    pathmunge /usr/sbin after  
    pathmunge /sbin after  
fi  
  
HOSTNAME=`bin/hostname 2>/dev/null`  
HISTSIZE=1000  
if [ "$HISTCONTROL" = "ignoreSpace" ]; then  
    export HISTCONTROL=ignoreboth  
else  
    export HISTCONTROL=ignoredups  
fi  
  
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL  
  
# By default, we want umask to get set. This sets it for login shell  
# Current threshold for system reserved uid/gids is 200  
# You could check uidgid reservation validity in  
# /usr/share/doc/setup-*/uidgid file  
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then  
    umask 002  
else  
    umask 022  
fi  
  
for i in /etc/profile.d/*.sh ; do  
    if [ -r "$i" ]; then  
        if [ "${#i}" != "$-" ]; then  
            . "$i"  
        else  
            . "$i" >/dev/null 2>&1  
        fi  
    fi  
done  
  
unset i  
unset pathmunge  
export PATH=/opt/FriendlyARM/toolschain/4.4.3/bin/:$PATH
```

环境变量添加成功。

执行# source /etc/profile 使环境变量生效。

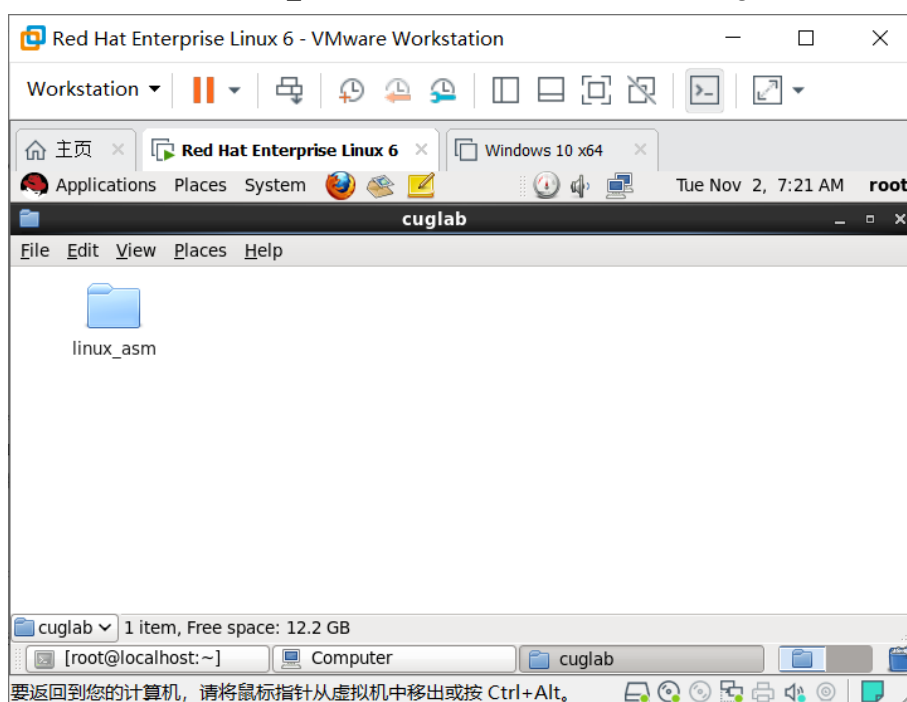
执行# echo \$PATH, 查看环境变量是否生效。

执行# arm-linux-gcc -v, 查看交叉编译工具的信息, 如下图:

```
[root@localhost ~]# vi /etc/profile
[root@localhost ~]# source /etc/profile
[root@localhost ~]# echo $PATH
/opt/FriendlyARM/toolschain/4.4.3/bin:/opt/FriendlyARM/toolschain/4.4.3/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/root/bin
[root@localhost ~]# arm-linux-gcc -v
Using built-in specs.
Target: arm-none-linux-gnueabi
Configured with: /opt/FriendlyARM/mini2440/build-toolschain/working/src/gcc-4.4.3/configure --build=i386-build_redhat-linux-gnu --host=i386-build_redhat-linux-gnu --target=arm-none-linux-gnueabi --prefix=/opt/FriendlyARM/toolschain/4.4.3 --with-sysroot=/opt/FriendlyARM/toolschain/4.4.3/arm-none-linux-gnueabi/sys-root --enable-languages=c,c++ --disable-multilib --with-arch=armv4t --with-cpu=arm920t --with-tune=arm920t --with-float=soft --with-pkgversion=ctng-1.6.1 --disable-sjlj-exceptions --enable-cxa-atexit --with-gmp=/opt/FriendlyARM/toolschain/4.4.3 --with-mpfr=/opt/FriendlyARM/toolschain/4.4.3 --with-ppl=/opt/FriendlyARM/toolschain/4.4.3 --with-cloog=/opt/FriendlyARM/toolschain/4.4.3 --with-mpc=/opt/FriendlyARM/toolschain/4.4.3 --with-local-prefix=/opt/FriendlyARM/toolschain/4.4.3/arm-none-linux-gnueabi/sys-root --disable-nls --enable-threads=posix --enable-symvers=gnu --enable-c99 --enable-long-long --enable-target-optspace
Thread model: posix
gcc version 4.4.3 (ctng-1.6.1)
[root@localhost ~]#
```

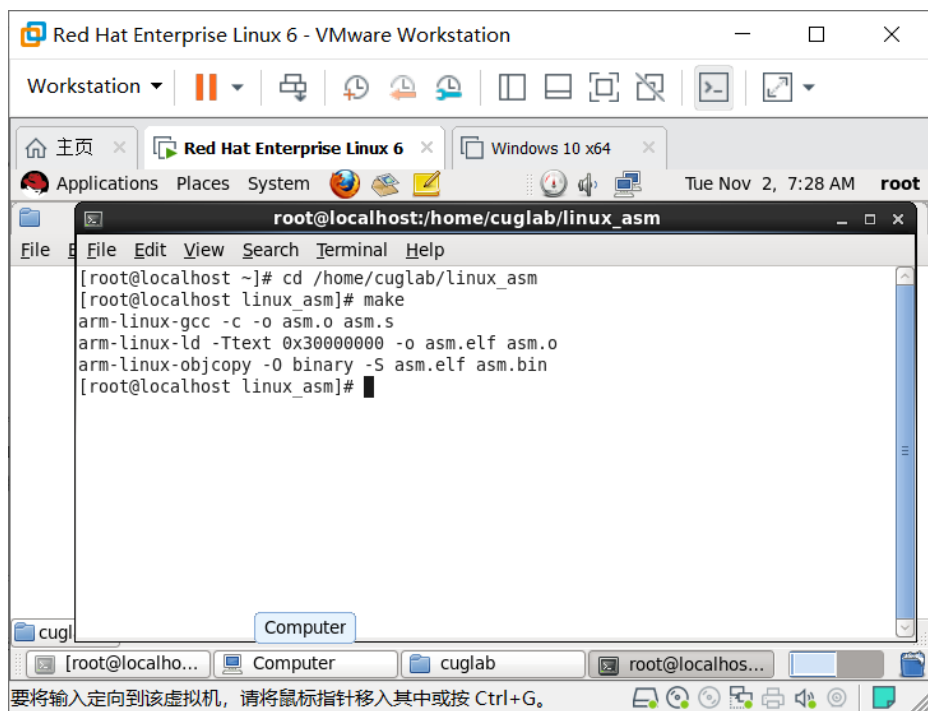
3.5 实验十 linux 环境下裸机程序设计及编译

1) 把本实验程序文件夹 linux_asm 复制到 Linux 系统目/home/cuglab 下:

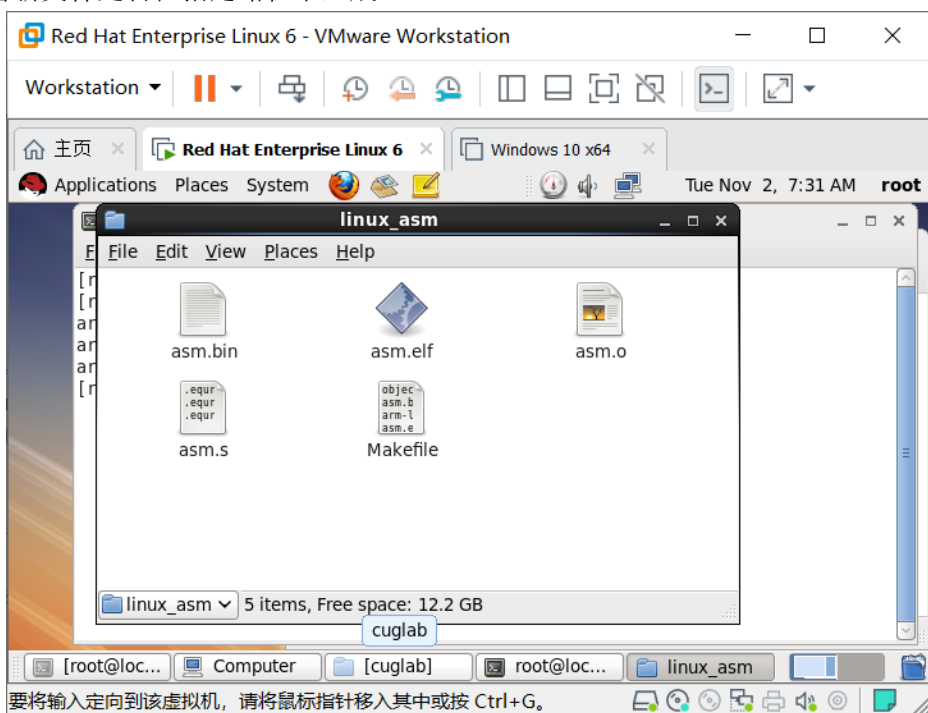


2) 执行命令:

```
# cd /home/cuglab/linux_asm
# make
```



3) 查看新文件是否在指定路径下生成:



文件生成成功。

第三篇 Linux 操作系统移植与驱动初体验

一、实验目的

了解嵌入式 Linux 操作系统下驱动程序的加载及使用。

二、实验要求

三、实验内容

3.1 实验十一 Linux 系统移植初体验

1) 烧写 bootloader 程序

烧写 u-boot: 将启动开关拨到 nor 启动, 选择选项 1, 下载 u-boot.bin 到 nandflash, u-boot。下载完成后, 出现如下界面:

```
[1] Download u-boot or other bootloader to Nand Flash
[2] Download Eboot (eboot.nb0) to Nand Flash
[3] Download Linux Kernel (zImage.bin) to Nand Flash
[4] Download winCE NK.bin to Nand Flash
[5] Download CRAMFS image to Nand Flash
[6] Download YAFFS image (root.bin) to Nand Flash
[7] Download Program (uCOS-II or TQ2440_Test) to SDRAM and Run it
[8] Boot the system
[9] Format the Nand Flash
[0] Set the boot parameters
[a] Download User Program (eg: uCOS-II or TQ2440_Test)
[b] Download LOGO Picture (.bin) to Nand Flash
[l] Set LCD Parameters
[n] Enter TFTP download mode menu
[o] Download u-boot to Nor Flash
[r] Reboot u-boot
[t] Test Linux Image (zImage)
[q] quit from menu
Enter your selection: 1
USB host is connected. waiting a download.

Now, Downloading [ADDRESS:30000000h,TOTAL:239218]
RECEIVED FILE SIZE: 239218 (233KB/S, 1S)

NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x20000 -- 100% complete.
OK

NAND write: device 0 offset 0x0, size 0x3a668
writing data at 0x3a000 -- 100% complete.
239208 bytes written: OK
```

2) 烧写操作系统内核

烧写操作系统内核 zImage: 保持开关为 nor 启动, 选择选项 3, 下载 zImage.bin 到 nandflash, 内核下载完成后, 出现如下界面:


```

[1] Download u-boot or other bootloader to Nand Flash
[2] Download Eboot (eboot.nb0) to Nand Flash
[3] Download Linux Kernel (zImage.bin) to Nand Flash
[4] Download WinCE NK.bin to Nand Flash
[5] Download CRAMFS image to Nand Flash
[6] Download YAFFS image (root.bin) to Nand Flash
[7] Download Program (uCOS-II or TQ2440_Test) to SDRAM and Run it
[8] Boot the system
[9] Format the Nand Flash
[0] Set the boot parameters
[a] Download User Program (eg: uCOS-II or TQ2440_Test)
[b] Download LOGO Picture (.bin) to Nand Flash
[l] Set LCD Parameters
[n] Enter TFTP download mode menu
[o] Download u-boot to Nor Flash
[r] Reboot u-boot
[t] Test Linux Image (zImage)
[q] quit from menu
Enter your selection: 3
USB host is connected. waiting a download.

```

```

Now, Downloading [ADDRESS:30000000h,TOTAL:1923054]
RECEIVED FILE SIZE: 1923054 (625KB/s, 35)

```

```

NAND erase: device 0 offset 0x200000, size 0x300000
Erasing at 0x4e0000 -- 100% complete.
OK

```

```

NAND write: device 0 offset 0x200000, size 0x1d57e4

```

```

writing data at 0x3d5000 -- 100% complete.
1923044 bytes written: OK

```

3) 烧写根文件系统

文件系统 rootfs: 保持开关为 nor 启动, 选择选项 6, 下载 rootfs.bin 到 nandflash, 文件。系统的下载需要的时间稍长, 文件系统下载完成后, 出现如下界面:

```

[1] Download u-boot or other bootloader to Nand Flash
[2] Download Eboot (eboot.nb0) to Nand Flash
[3] Download Linux Kernel (zImage.bin) to Nand Flash
[4] Download WinCE NK.bin to Nand Flash
[5] Download CRAMFS image to Nand Flash
[6] Download YAFFS image (root.bin) to Nand Flash
[7] Download Program (uCOS-II or TQ2440_Test) to SDRAM and Run it
[8] Boot the system
[9] Format the Nand Flash
[0] Set the boot parameters
[a] Download User Program (eg: uCOS-II or TQ2440_Test)
[b] Download LOGO Picture (.bin) to Nand Flash
[l] Set LCD Parameters
[n] Enter TFTP download mode menu
[o] Download u-boot to Nor Flash
[r] Reboot u-boot
[t] Test Linux Image (zImage)
[q] quit from menu
Enter your selection: 6
USB host is connected. waiting a download.

```

```

Now, Downloading [ADDRESS:30000000h,TOTAL:58884682]
TQ2440 board's Nand is 256MB or 1GB, please choose correct yaffs image.

```

```

Image Length = 58884672, StartAddr = 0x500000, Length = 0xfb00000.
writing data at 0xda59800 -- 85% complete.
skipping bad block 0xdb0000
writing data at 0x10000000 -- 100% complete.
Yaffs Images writed to Nand Flash complete!

```

4) 启动操作系统

这样就将 linux 系统通过 usb 下载到开发板了, 将开发板启动开关拨到 nand, 启动开

发板。

5) 在超级终端看到系统正常启动起来。如下图所示：

```
[root@EmbedSky ~]# ls
bin          linuxrc      root         tq2440_serial0
dev          lost+found  sbin        udisk
etc          mnt         sddisk      usr
home        opt         sys         var
lib         proc        tmp         web
```

3.2 实验十一 Linux 操作系统下驱动开发初体验

1) 将所给源码中的三个文件拷贝到 U 盘

将 EmbedSky_hello.ko、EmbedSky_leds.ko、leds 拷贝到 u 盘。其中 EmbedSky_hello.ko 是 hello world 驱动程序，EmbedSky_leds.ko 是 LED 灯的驱动程序，leds 是应用程序（程序可以通过调用 LED 灯驱动程序来控制亮灭）。

```
leds 2019/11/13 14:18 文件 4 KB
```

2) 将 U 盘插入开发板的 usb 接口

将 U 盘插入开发板的 usb 接口，如下所示：

```
[root@EmbedSky ~]#
```

3) 在开发板 Linux 下查询 U 盘文件

切换到 udisk 目录，如果 U 盘加载成功，可以查到到拷贝到 U 盘里面的文件。

```
EmbedSky_hello.ko EmbedSky_leds.ko leds
```

4) 加载 hello 驱动程序（硬件无关）

加载第一个驱动程序，命令 insmod EmbedSky_hello.ko，终端打印如下：

```
EmbedSky_hello.ko
```

5) 卸载驱动程序

卸载该驱动程序，命令 rmmod EmbedSky_hello.ko，终端打印如下：

```
Exit!
```

6) 加载 LED 驱动程序（硬件相关）

加载第二个驱动程序，命令 insmod EmbedSky_leds.ko，终端打印如下：

```
tq2440-leds initialized
```

7) 查看设备节点

查看设备节点，命令 ls /dev/tq2440-leds，终端打印如下：

```
/dev/tq2440-leds
```

8) 测试应用程序

运行 leds 需要给它传递两个参数，其执行格式为：

```
./leds
led_no
```

0|1

其中参数“led_no”表示 led 灯编号，四个 led 分别对应 1、2、3、4，当该参数为 5 表示同时控制四个灯；参数“0|1”表示控制 led 亮(1)灭(0)。

四、思考题

1、思考 windows 环境下与 Linux 环境下开发裸机程序的区别有什么？

windows 环境下在 ads 中编写完代码之后，需要自行修改 DebugRel Settings 设置，通过编译后，利用 AXD 软件进行调试。然后将生成的 .bin 文件通过串口下载到开发板上运行。Linux 环境下编写完代码后，可以通过执行 makefile 文件编译代码，生成的 .bin 文件可以移至 windows 环境，再通过串口下载到开发板。

2、make 及 makefile 的作用是什么？

make 命令会读取 makefile 文件的内容，它先确定目标文件和要创建的文件，然后比较该目标所依赖的源文件的日期和时间以决定该采取那条规则来构建目标。通常在创建最终的目标文件之前，它需要先创建一些中间目标。make 命令会根据 makefile 文件来确定目标文件的创建顺序以及正确的规则调用顺序。

makefile 文件和 make 工具一起使用，用于控制工程项目的编译和链接，也可以用来编写手册页和程序的安装。makefile 文件中通常包含源文件和目标文件的依赖关系以及从源文件生成目标文件的规则。

3、第二篇实验与第三篇实验的 Linux 系统一样吗？如不同，不同之处有什么？

不一样。第二篇实验是将 Linux 下交叉编译生成的 bin 文件拷贝到 windows 操作系统下，使用串口超级终端将文件下载到开发板上；而第三篇实验是将嵌入式 Linux 系统移植到 ARM 开发板上，并基于嵌入式 Linux 操作系统直接实现目标驱动程序的加载及使用。

4、关于 Linux 操作系统，你还想要了解什么？

Linux 相对于 Windows 系统的优越之处与不足之处；目前国内 Linux 的主要应用领域及发展趋势等。

5、关于嵌入式 Linux 操作系统的开发，你还希望学习什么？

学习标准 I/O 库、多任务编程中的多进程和多线程等，提升对 Linux 应用开发的理解和代码调试的能力。

五、心得与体会

这几次实验相对较简单，按照实验指导书，熟悉一些 linux 的命令之后基本可以完成，这也是我第一次尝试 linux 操作系统的移植。实验指导书十分详细，资料也比较齐全，但也由于许多驱动程序、代码均是老师提供的，所以在有些操作时我只知道怎么做，却不太能理解为什么这样做，希望在后续嵌入式实习时可以对 linux 系统的移植和使用有更加深刻的体会。