**《嵌入式操作系统》实验指导书**

内蒙古工业大学

计算机系

2022年10月

说 明

各小组在完成每一个实验后要找指导老师进行验收、登记；实验完成后要提交实验报告，实验报告封面参见附1，实验报告格式要求参见附2。

## 实验一 Linux内核移植与编译实验

**1. 实验目的**

* 了解 Linux 内核相关知识与内核结构
* 了解 Linux 内核在 ARM 设备上移植的基本步骤和方法
* 掌握 Linux 内核裁剪与定制的基本方法

**2. 实验内容**

* 分析 Linux 内核的基本结构，了解 Linux 内核在 ARM 设备上移植的一些基本步骤及常识。
* 学习 Linux 内核裁剪定制的基本配置方法，利用QEMU模拟vexpress\_ca9x4开发板，对Linux 内核进行自定义功能(如helloworld 显示)的添加，并重新编译内核源码，生成内核文件，下载到vexpress\_ca9x4开发板中测试。

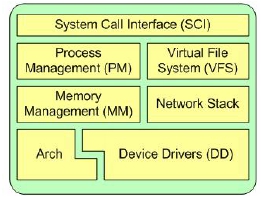
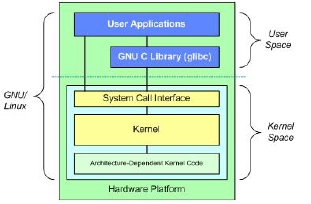
**3. 实验环境**

* 硬件：PC机
* 软件：Vmware Workstation +UBUNTU 22.04 LTS + ARM-LINUX交叉编译开发环境 + QEMU

**4. 实验原理**

**4.1 Linux 内核背景知识**

内核是操作系统的核心，具有很多最基本功能，如虚拟内存、多任务、共享库、需求加载、共享的写时拷贝(copy-on-write)可执行程序和TCP/IP 网络功能。如下图所示：



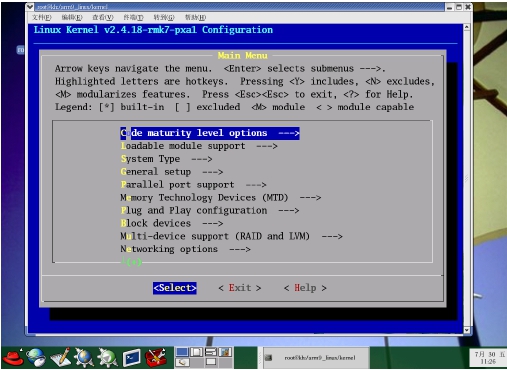
系统结构框图

上图中最上面是用户（或应用程序）空间。这是用户应用程序执行的地方。用户空间之下是内核空间，Linux 内核正是位于这里。

**4.2 Linux 内核配置及裁剪**

Linux 内核的编译菜单有好几个版本，运行：

* make config：命令行配置，界面方式不友好。
* make menuconfig：基于光标库运行，界面友好。
* make xconfig：基于QT运行库。



“make menuconfig”在选择相应的配置时，有三种选择方式，它们分别代表的含义如下：

Y－-将该功能编译进内核

N－-不将该功能编译进内核

M－-将该功能编译成可以在需要时动态插入到内核中的模块

需要使用空格键进行选取。在每一个选项前都有一个括号， 有的是中括号有的是尖括号，还有圆括号。用空格键选择时可以发现，中括号里要么是空，要么是"\*"，而尖括号里可以是空，"\*"和"M"这表示前者对应的项要么不要，要么编译到内核里；后者则多一样选择，可以编译成模块。而圆括号的内容是要你在所提供的几个选项中选择一项。

**5. 实验步骤**

**5.1 基本软件安装和环境配置**

（1）基本软件安装

sudo apt-get install qemu

sudo apt-get install qemu-system-arm

sudo apt-get install gcc-arm-linux-gnueabi

sudo apt install gcc

sudo apt install make

sudo apt-get install build-essential

sudo apt-get install libncurses5

sudo apt-get install libncurses5-dev

sudo apt-get install u-boot-tools

（2）配置tftp

下载tftp：sudo apt-get install tftp-hpa tftpd-hpa

修改tftp配置文件（主要是修改tftp的发布目录，自己定义）：sudo vi /etc/default/tftpd-hpa

修改内容如下：

# /etc/default/tftpd-hpa

TFTP\_USERNAME="tftp"

#修改成自己建立的tftp发布目录；当前登录普通用户的主目录位于：/home/账户名称

TFTP\_DIRECTORY="/home/zz/tftpboot"

TFTP\_ADDRESS=":69"

TFTP\_OPTIONS="--secure"

创建tftp的发布目录：sudo mkdir ~/tftpboot

修改tftp的发布目录权限：sudo chmod 777 ~/tftpboot

重启tftp服务器来让其生效：sudo /etc/init.d/tftpd-hpa restart

（3）配置nfs

安装nfs服务：sudo apt-get install nfs-kernel-server

创建nfs目录：mkdir ~/nfs

配置nfs：vi /etc/exports

在文件的最后一行添加如下内容：(/home/zz是zz用户的主目录，根据自己建立的nfs目录修改)

/home/zz/nfs \*(rw,nohide,insecure,no\_subtree\_check,async,no\_root\_squash)

重启nfs服务：sudo /etc/init.d/nfs-kernel-server restart

（4）配置qemu和ubuntu网络桥接【可选：这一步骤先略过】

安装工具：

sudo apt-get install uml-utilities

sudo apt-get install bridge-utils

sudo apt-get install ifupdown

打开相应的文件进行配置网络，为qemu添加一块网卡br0：sudo vi /etc/network/interfaces

添加（需要注意ens33为当前网卡，如果你的默认网卡是eth0，就要将ens33修改成eth0）：

auto br0

iface br0 inet dhcp

# iface br0 inet static

# address 192.168.0.1

# netmask 255.255.255.0

# gateway 192.168.0.254

bridge\_ports ens33

bridge\_fd 9

bridge\_hello 2

bridge\_maxage 12

bridge\_stp off

# The tap0 network interface(s)

auto tap0

iface tap0 inet manual

# iface tap0 inet static

# address 192.168.0.2

# netmask 255.255.255.0

# gateway 192.168.0.254

pre-up tunctl -t tap0 -u root

pre-up ifconfig tap0 0.0.0.0 promisc up

post-up brctl addif br0 tap0

重启ubuntu：sudo init 6

（5）关闭和禁用防火墙

sudo ufw disable

sudo iptables -F

**5.2 u-boot建立**

下载u-boot源代码文件进行解压（可在用户主目录中操作）：tar -xjvf u-boot-2018.01-rc1.tar.bz2

进入u-boot文件夹：cd u-boot-2018.01-rc1/

添加环境变量及启动参数：vi include/configs/vexpress\_common.h

加入以下内容(CONFIG\_SERVERIP是ubuntu IP，同学们根据你的实际情况进行修改)：

#define CONFIG\_IPADDR 10.0.2.15

#define CONFIG\_NETMASK 255.255.255.0

#define CONFIG\_SERVERIP 192.168.254.142

#define CONFIG\_GATEWAYIP 10.0.2.3

修改Makefile：vi Makefile

打开文件做如下更改：

ifeq ($(HOSTARCH),$(ARCH))

CROSS\_COMPILE ?=arm-linux-gnueabi-

endif

CROSS\_COMPILE ?=arm-linux-gnueabi-

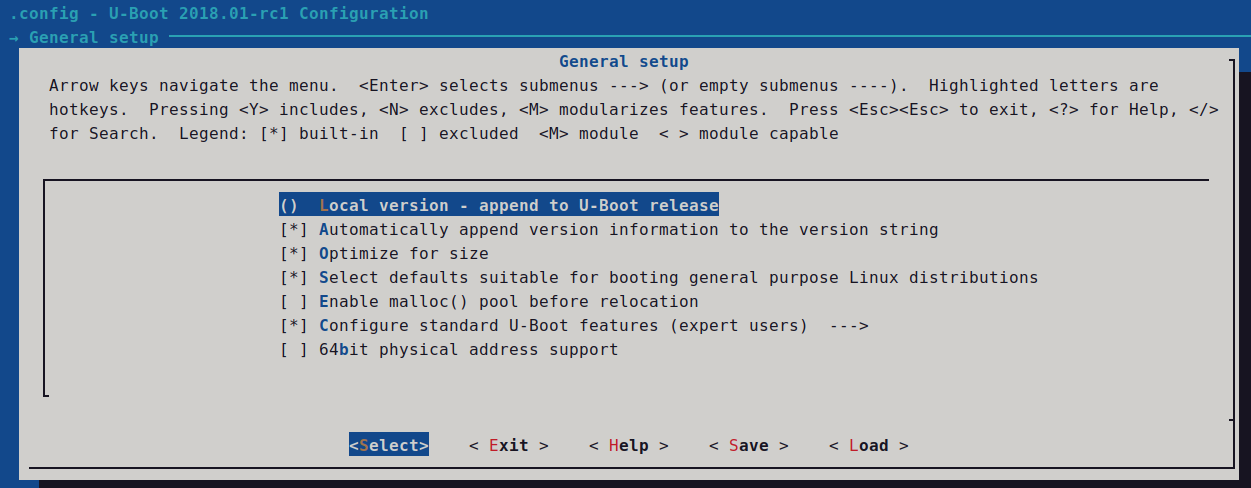
配置u-boot，执行：

export ARCH=arm

make vexpress\_ca9x4\_defconfig

make menuconfig

在menuconfig中，如下图，添加自己的信息；注意添加之后要保存当前配置（保存成.config）



编译，执行：make -j4

**5.3 建立测试脚本运行qemu，测试u-boot（在u-boot根目录下）**

建立测试脚本：vi ./run.sh

添加如下内容：

#有GUI的

#qemu-system-arm -M vexpress-a9 -m 512M -kernel u-boot -nic user

#无GUI的

qemu-system-arm -M vexpress-a9 -nographic -m 512M -kernel u-boot -nic user

修改权限：chmod +x ./run.sh

运行qemu：sudo ./run.sh

【启动u-boot后按下任意键进入下载模式，可以运行u-boot命令进行测试；如打印、设置、保存环境变量，ping主机等。请参见课件PPT。】

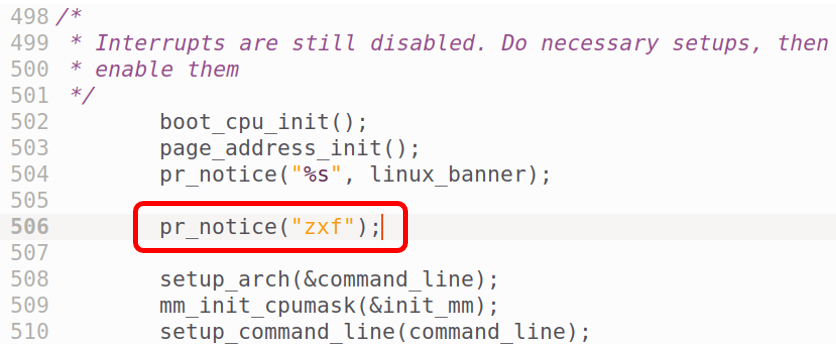
**5.4 建立linux内核**

（1）下载内核源代码linux-4.9.48解压（可在用户主目录中操作）：tar -xzvf linux-4.9.48.tar.gz

（2）进入内核源代码根目录：cd linux-4.9.48

（3）添加版本注释信息：打开文件“init/main.c”,在“start\_kernel”函数中，**添加自己名字拼音**，如下图：

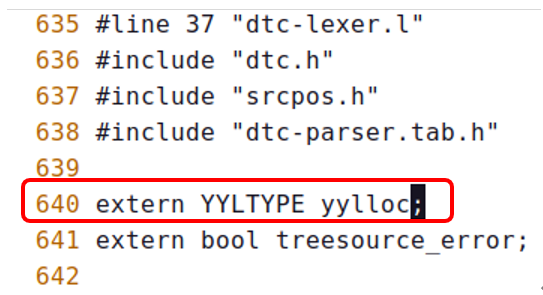
vi init/main.c



（4）修改bug，高版本的gcc编译内核出现的问题，解决方法：

进入内核的dtc目录(scripts/dtc)，修改文件dtc-lexer.lex.c\_shipped，在“YYLTYPE yylloc;”前增加extern(该修改项在640行):

vi scripts/dtc/dtc-lexer.lex.c\_shipped



（5）编写简单的测试驱动（内核）程序 helloworld.c 并修改内核目录中相关文件，添加对测试驱动程序的支持。

使用vi编辑器在drivers/char/下手动编写实验代码helloworld.c：vi drivers/char/helloworld.c

程序如下：

#include <linux/init.h>

#include <linux/module.h>

MODULE\_LICENSE("Dual BSD/GPL");

//驱动程序入口函数

static int hello\_init(void)

{

printk(KERN\_ALERT "##############Hello, world############\n");

return 0;

}

//驱动程序出口函数

static void hello\_exit(void)

{

printk(KERN\_ALERT "###############Goodbye, world#########\n");

}

module\_init(hello\_init);

module\_exit(hello\_exit);

有关驱动程序的编写规范，请参考课程教材，本实验只在编写简单的驱动(内核)程序并加入到Linux内核目录树中，使同学们熟悉编译内核的过程。该驱动程序是向终端输出相关程序信息。进入实验内核源码目录修改drivers/char/目录下的Kconfig文件，按照Kconfig语法添加helloworld程序的菜单支持：

vi drivers/char/Kconfig

在Kconfig文件中添加如下：

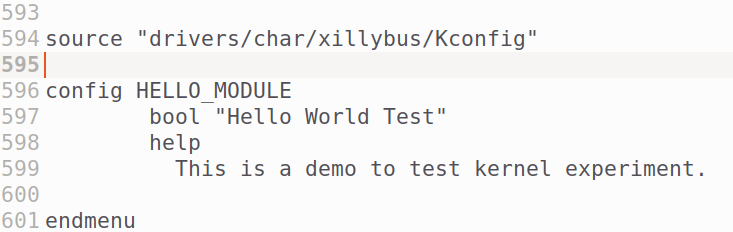
config HELLO\_MODULE

bool "Hello World Test"

help

This is a demo to test kernel experiment.

如下图：



注意，config HELLO\_MOULDE 段要与前后段有空格隔开，且bool等变量要与行开头有 TAB 符号位隔开。注意 Kconfig 的格式(可以拷贝原有内容进行修改)。

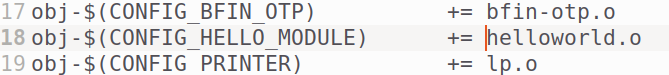
进入实验内核源码目录修改drivers/char/目录下的 Makefile 文件，按照内核中 Makefile 语法添加 helloworld程序的编译支持：

vi drivers/char/Makefile

在 Makefile 中添加如下一行：

**obj-$(CONFIG\_HELLO\_MODULE) += helloworld.o**

如下图：



（6）配置内核，执行：

export ARCH=arm

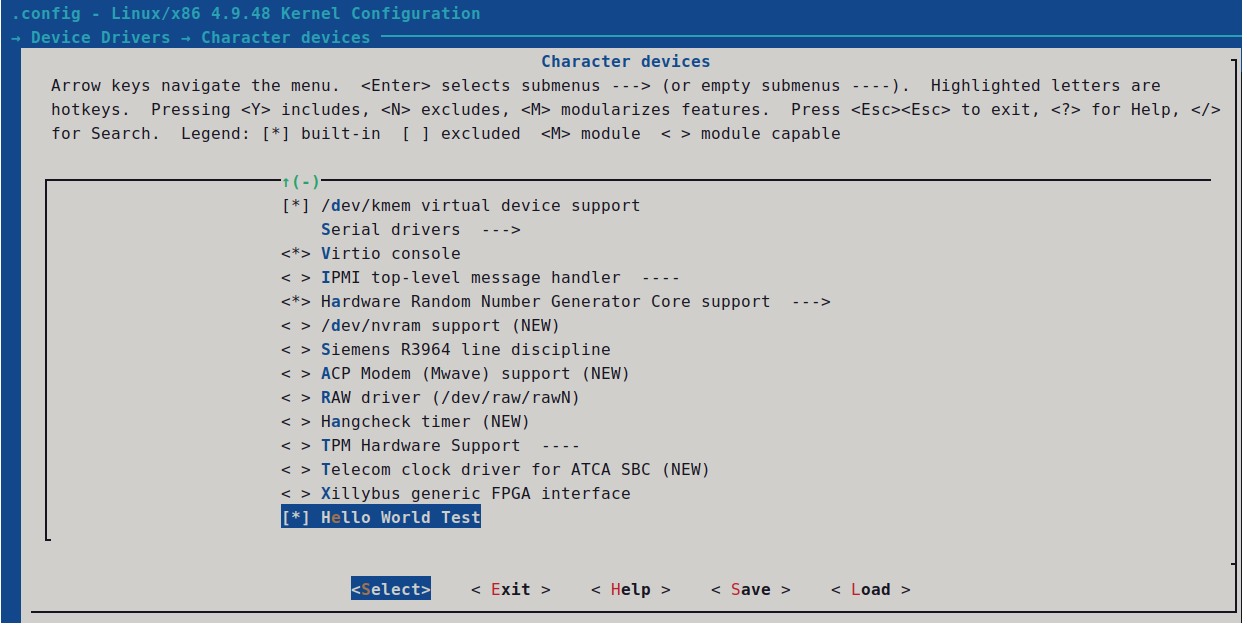
export CROSS\_COMPILE=arm-linux-gnueabi-

make vexpress\_defconfig

运行 make menuconfig 配置内核对 helloworld 程序的支持：

make menuconfig

进入到 Device Drivers --->菜单，再进入到 Character devices--->如图：



进入该菜单会发现[ ] Hello World Test 选项，按下空格将其静态编译进内核。

退出前保存内核配置**（注意：退出前要将内核配置保存成 .config 文件）**

（7）编译内核，执行：

make zImage -j4

make LOADADDR=0x60003000 uImage -j4

make modules -j4

make dtbs

之后在Linux内核源码的arch/arm/boot目录下生成了内核映像文件uImage和zImage，在Linux内核源码的上一层目录建立一个目录armbin，然后将内核文件uImage和zImage及vexpress-v2p-ca9.dtb，5.2节生成的u-boot拷贝到armbin中，将uImage和zImage及vexpress-v2p-ca9.dtb拷贝到tftp发布目录（tftp发布目录同学们要根据自己的情况修改下面的cp命令）：

cd ..

mkdir armbin

cp linux-4.9.48/arch/arm/boot/zImage armbin/

cp linux-4.9.48/arch/arm/boot/uImage armbin/

cp linux-4.9.48/arch/arm/boot/dts/\*ca9.dtb armbin/

cp u-boot-2018.01-rc1/u-boot ./armbin/

cp armbin/uImage ~/tftpboot/

cp armbin/vexpress-v2p-ca9.dtb ~/tftpboot/

在armbin目录中建立测试脚本，qemu中运行u-boot，然后通过tftp下载Linux内核到内存中，再启动Linux（本测试需要用到根文件系统，要提前将老师提供的根文件系统a9rootfs.ext3拷贝到armbin目录中）：

cd armbin

vi run1.sh

输入以下内容：

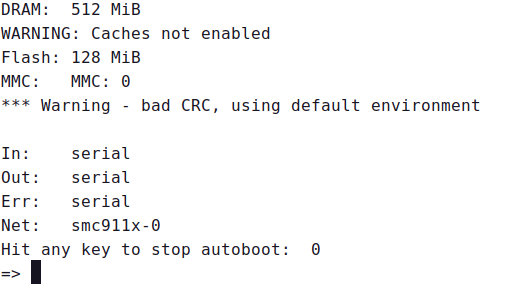
qemu-system-arm -M vexpress-a9 -nographic -m 512M -kernel u-boot -sd a9rootfs.ext3 -nic user

保存，运行qemu：

chmod +x run1.sh

./run1.sh

启动u-boot后按下任意键进入下载模式，如下图所示：



输入命令启动Linux内核（注意以下命令是在qemu运行的虚拟机中输入的）：

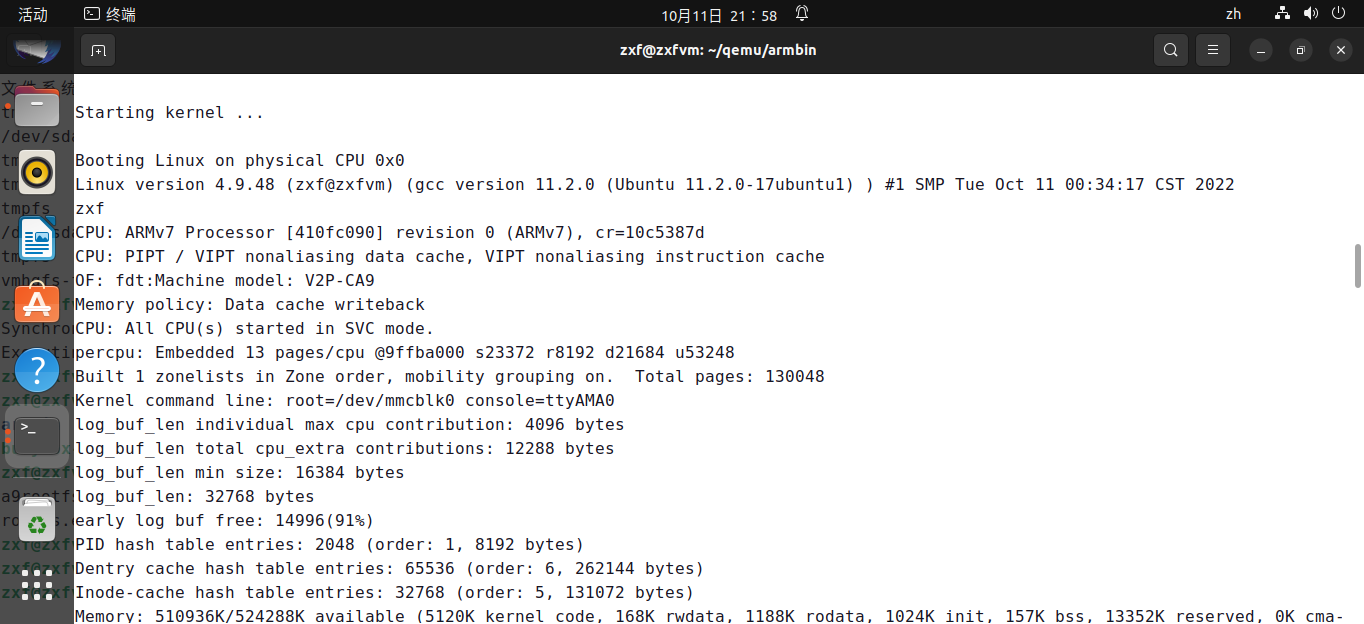
tftp 0x60003000 uImage

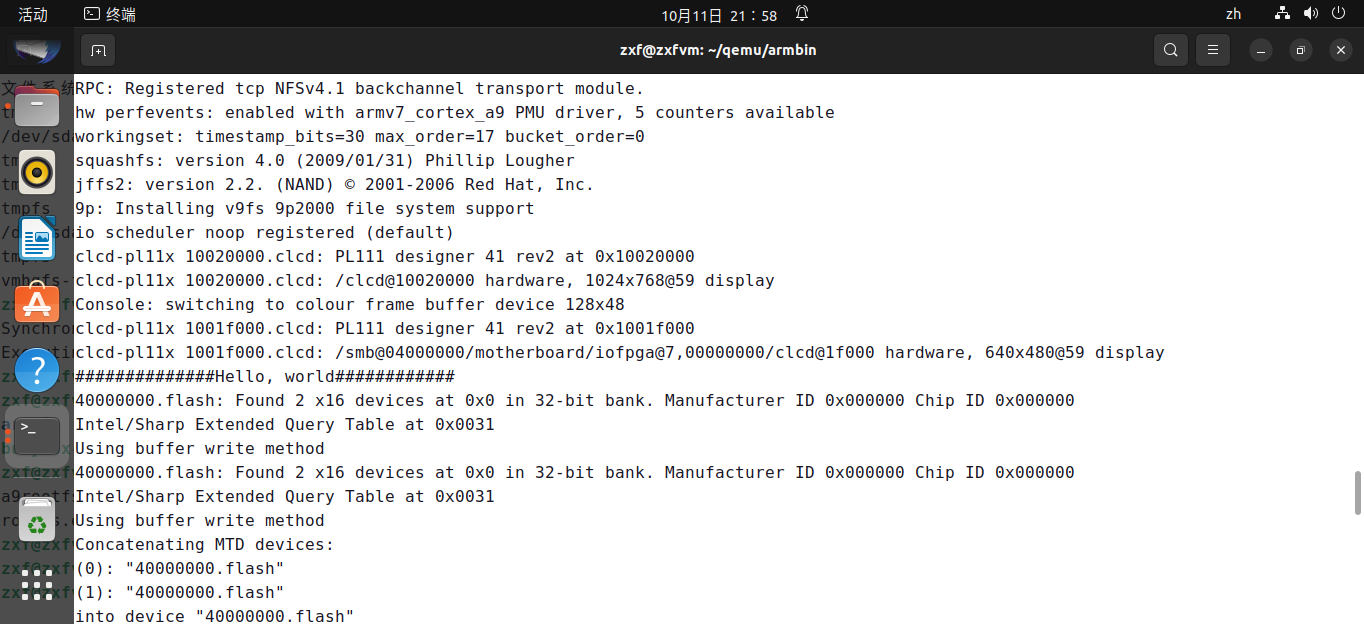
tftp 0x60500000 vexpress-v2p-ca9.dtb

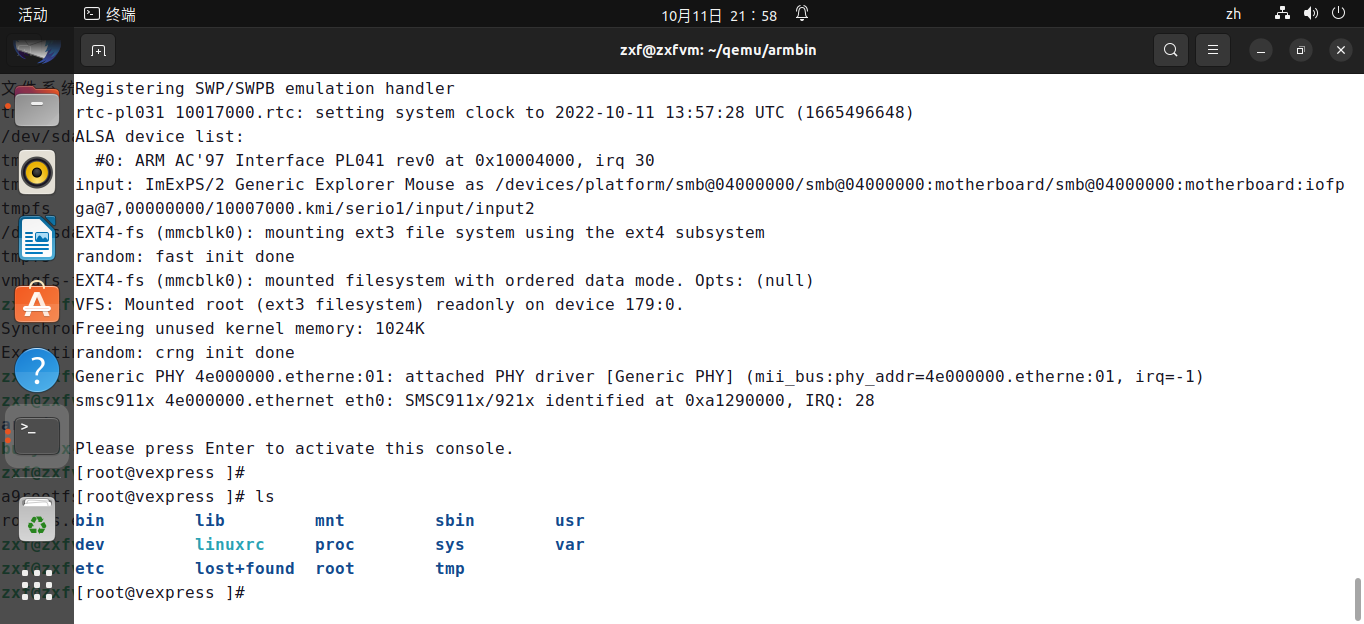
setenv bootargs 'root=/dev/mmcblk0 console=ttyAMA0'

bootm 0x60003000 - 0x60500000

启动Linux内核后，如下图：

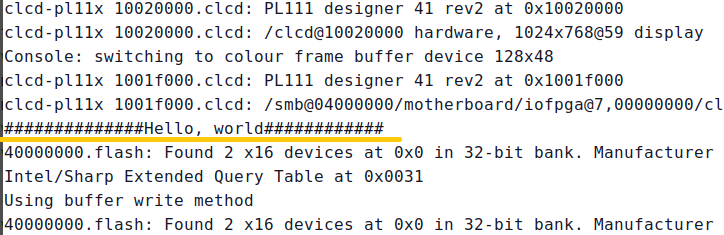






同学们需要在Linux内核启动输出的信息中找出：自己添加的名字拼音和helloworld程序的输出信息，如：





注1：虚拟机和主机共享文件设置：

（1）在VMware设置界面中，设置共享目录

（2）如果在虚拟机中无法访问/mnt/hgfs，则执行下面的命令：

sudo chmod 777 /mnt/hgfs

vmhgfs-fuse .host:/ /mnt/hgfs

## 实验二 根文件系统实验

**1. 实验目的**

* 了解Linux系统下根文件系统结构
* 掌握根文件系统的搭建过程
* 掌握busybox等工具的使用方法

**2. 实验内容**

* 使用 busybox 生成文件系统中的命令部分，制作根文件系统。
* 分析根文件系统etc目录下重要配置文件的格式及语法，熟悉根文件系统的启动过程。

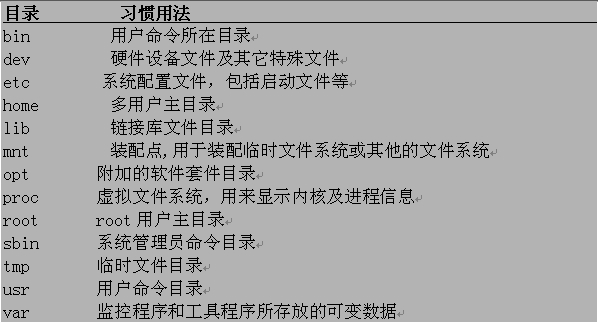
**3. 实验环境**

* 硬件：PC机
* 软件：Vmware Workstation +UBUNTU 22.04 LTS + ARM-LINUX交叉编译开发环境 + QEMU

**4. 实验原理**

**4.1 嵌入式根文件系统**

Linux下的根文件系统目录结构如下。



对于用途单一的嵌入式系统，上边的一些用于多用户的目录可以省略，例如/home、/opt目录等。而/bin、/dev、/etc、/lib、/sbin和/usr目录，是几乎每个系统必备的目录，也是不可或缺的目录。

**4.2 Busybox 工具集**

小型的嵌入式 Linux 系统制作 root 根文件系统时有一个常用的利器：BusyBox。Busybox 是 Debian

GNU/Linux 的大名鼎鼎的 Bruce Perens 首先开发，使用在 Debian 的安装程序中。后来又有许多 Debian

developers 贡献力量，这其中尤推 busybox 目前的维护者 Erik Andersen，他患有癌症，可是却是一名优秀的自由软件开发者。

Busybox 编译出一个单个的独立执行程序，就叫做 busybox。但是它可以方便的进行配置，执行 ash shell的功能，以及几十个各种小应用程序的功能。这其中包括有一个迷您的 vi 编辑器，以及其他诸如 sed, ifconfig,mkdir, mount, ln, ls, echo, cat ... 等等这些都是一个正常的系统上必不可少的工具，但是如果我们把这些程序的原件拿过来的话，它们的体积加在一起，让人吃不消。可是 busybox 有全部的这么多功能，大小也不过100K 左右。而且，用户还可以根据自己的需要，决定到底要在 busybox 中编译进哪几个应用程序的功能。这样的话，busybox 的体积就可以进一步缩小了。Busybox 的具体配置和编译将在实验部分介绍。

**5. 实验步骤**

（1）下载busybox源代码：busybox-1.34.1.tar.bz2

（2）解压busybox-1.34.1.tar.bz2，进入busybox目录进行配置：

tar -xjvf busybox-1.34.1.tar.bz2

cd busybox-1.34.1

修改Makefile：vi Makefile

如下：

ARCH ?= arm

CROSS\_COMPILE=arm-linux-gnueabi-

配置：【make menuconfig中设置之后保存配置文件为.config】

make defconfig

make menuconfig

make

make install

可以看到\_install 目录下生成了根文件系统常用的命令及工具，接下来的工作，可以根据需要将\_install目录下生成的命令拷贝到根文件系统相应目录下。

（3）创建rootfs目录：

mkdir ../rootfs

（4）拷贝busybox命令：

cp \_install/\* -r ../rootfs/

mkdir -p ../rootfs/{lib,proc,sys,tmp,root,var,mnt}

（5）从工具链中拷贝运行库到lib下：

cp -P /usr/arm-linux-gnueabi/lib/\* ../rootfs/lib/

rm ../rootfs/lib/\*.a

将etc.tar.gz拷贝到rootfs的上层目录，解压，拷贝配置文件：

cd ..

tar -xzvf etc.tar.gz

cp etc rootfs/ -arf

（6）创建4个tty终端设备

c代表字符设备，4是主设备号，1~2~3~4是次设备号

mkdir -p rootfs/dev/

sudo mknod rootfs/dev/tty1 c 4 1

sudo mknod rootfs/dev/tty2 c 4 2

sudo mknod rootfs/dev/tty3 c 4 3

sudo mknod rootfs/dev/tty4 c 4 4

sudo mknod rootfs/dev/console c 5 1

sudo mknod rootfs/dev/null c 1 3

（7）生成镜像

dd if=/dev/zero of=rootfs.ext3 bs=1M count=32

（8）格式化生成ext3文件系统

mkfs.ext3 rootfs.ext3

（9）先挂载刚才的镜像然后将文件拷贝到镜像中

mkdir tmpfs

sudo mount -t ext3 rootfs.ext3 tmpfs/ -o loop

sudo cp -r rootfs/\* tmpfs/

sudo umount tmpfs

cp rootfs.ext3 ~/armbin

（10）测试运行

在~/armbin目录，建立脚本文件run2.sh：

cd armbin

vi run2.sh

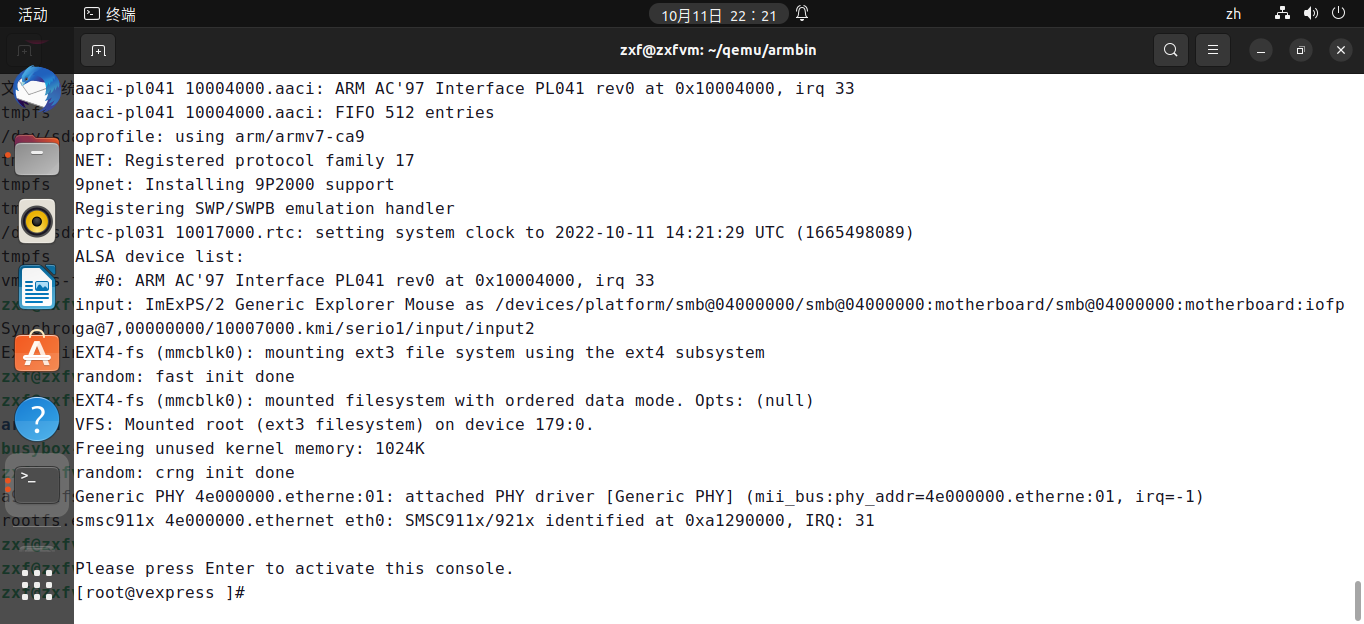
内容如下：

qemu-system-arm -M vexpress-a9 -m 512M -dtb vexpress-v2p-ca9.dtb -kernel zImage -nographic -append "root=/dev/mmcblk0 rw console=ttyAMA0" -sd rootfs.ext3 -net nic -net user,hostfwd=tcp::8080-:8000

运行：

chmod +x run2.sh

./run2.sh



（11）挂载主机上的NFS共享目录【提前需要配置nfs，nfs路径及IP地址需要大家修改成自己的】

mount -t nfs -o nolock 10.0.2.2:/home/zxf/nfs /tmp

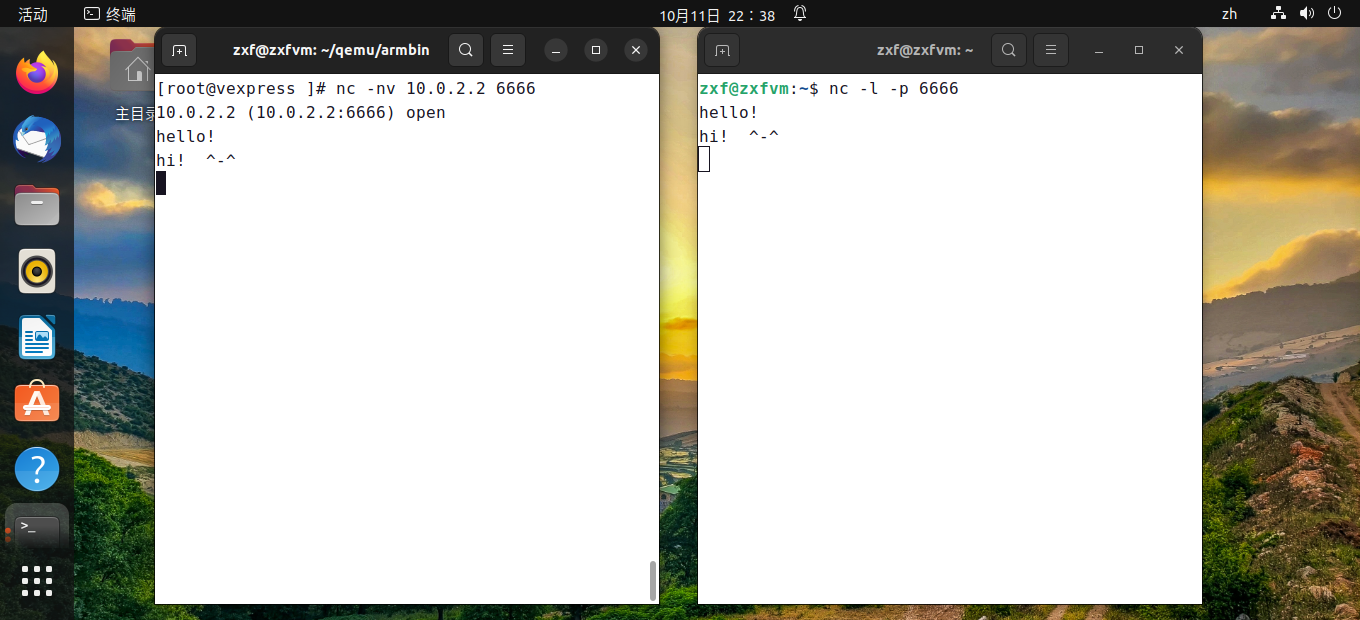
大家可以在NFS共享目录放入程序，在主机上交叉编译，再在QEMU虚拟机中运行测试。

（12）nc测试：

Ubuntu主机运行：nc -l -p 6666

qemu虚拟机中运行：nc -nv 10.0.2.2 6666

然后可以在Ubuntu主机或qemu虚拟机中输入任何文字信息，进行聊天测试，如下图：



## 实验三 模块方式驱动实验

**1. 实验目的**

* 学习在 LINUX 下进行驱动设计的原理
* 掌握使用模块方式进行驱动开发调试的过程
* 熟悉 Linux 系统下驱动编程
* 编程实现一个虚拟温湿度传感器的驱动程序

**2. 实验内容**

* 编程实现一个虚拟温湿度传感器的驱动程序及应用测试程序

**3. 实验环境**

* 硬件：PC机
* 软件：Vmware Workstation +UBUNTU 22.04 LTS + ARM-LINUX交叉编译开发环境 + QEMU

**4. 实验原理**

请参见课程教材关于“ARM Linux字符设备驱动开发”部分。

**5. 实验步骤**

（1）程序要求

编写一个虚拟温湿度传感器的驱动程序及应用测试程序并进行调试，驱动程序完成如下功能：

* 在内核模块初始化函数中，实现动态申请设备号并注册该设备，动态建立设备文件节点；
* 在内核模块的退出函数中，实现设备注销，删除该设备文件节点；
* 实现file\_operations设备驱动文件结构体中的read函数，完成如下功能：
* 随机生成温度值和湿度值；为用户应用程序提供采集接口，用户应用程序可以通过此接口读出虚拟的温度值和湿度值。
* 实现file\_operations设备驱动文件结构体中的open和release函数功能，其中：
* open函数：设置一个计数器count，每调用一次open函数，计数器count加1。
* release函数：每调用一次release函数，计数器count减1。

（2）程序运行

以上驱动程序和测试应用程序编写好后，进行交叉编译，如驱动程序名称为sht11.c；测试应用程序名称为sht11test.c；则编写Makefile文件，进行交叉编译，步骤参考如下：

vi Makefile

#Makefile文件内容（参考）：

TARGET = sht11test

CROSS\_COMPILE = arm-linux-gnueabi-

CC = $(CROSS\_COMPILE)gcc

ifeq ($(KERNELRELEASE),)

# /home/zxf/linux-4.9.48 为内核源码路径，请根据自己的情况修改

KERNELDIR ?=/home/zxf/linux-4.9.48

PWD := $(shell pwd)

all: $(TARGET) modules

$(TARGET):

$(CC) -o $(TARGET) $(TARGET).c

modules:

$(MAKE) -C $(KERNELDIR) M=$(PWD) modules

modules\_install:

$(MAKE) -C $(KERNELDIR) M=$(PWD) modules\_install

clean:

rm -rf \*.o \*~ core .depend .\*.cmd \*.ko \*.mod.c .tmp\_versions $(TARGET)

.PHONY: modules modules\_install clean

else

# called from kernel build system: just declare what our modules are

obj-m := sht11.o

endif

交叉编译：make

之后将sht11.ko及sht11test拷贝到NFS共享目录中，启动QEMU虚拟机，挂载NFS共享目录，加载驱动程序，最后运行测试应用程序，查看结果。步骤参考如下：

启动QEMU虚拟机后，挂载NFS共享目录命令（视情况修改NFS共享目录路径,以下命令在QEMU虚拟机中执行）：

mount -t nfs -o nolock 10.0.2.2:/home/zxf/nfs /mnt/nfs

进入NFS共享目录：cd /mnt/nfs

加载驱动：insmod sht11.ko

运行测试程序：./ sht11test

【备注1】

在QEMU ARM虚拟机上运行：

确认运行的QEMU ARM虚拟机上运行的Linux内核版本，然后下载对应版本的Linux内核源代码，解压，进行配置，编译。参考步骤如下：

（1）修改linux内核源码顶层目录下的Makefile：

将：

ARCH ?= $(SUBARCH)

CROSS\_COMPILE ?= $(CONFIG\_CROSS\_COMPILE:"%"=%)

修改成：

ARCH ?= arm

CROSS\_COMPILE ?= arm-linux-gnueabi-

（2）编译：

make vexpress\_defconfig

make zImage -j4

make modules -j4

（3）之后再修改内核模块（驱动）的Makefile中的KERNELDIR：

KERNELDIR:= 内核源码路径

之后再编译内核模块（驱动）即可。

【备注2】可以参考实验资料中的“testdriver.tar.gz”中驱动程序。

## 实验四 WEB服务器移植与模块驱动综合实验

**1. 实验目的**

* 了解常用的嵌入式Web服务器及其原理
* 掌握基本的CGI程序设计方法
* 熟悉字符设备驱动程序设计

**2. 实验内容**

* 编写CGI程序，调用多个设备模块驱动
* 在嵌入式Web服务器httpd上运行动态网页，运行测试CGI程序

**3. 实验环境**

* 硬件：PC机
* 软件：Vmware Workstation +UBUNTU 22.04 LTS + ARM-LINUX交叉编译开发环境 + QEMU

**4. 实验原理**

随着Internet技术的兴起，在嵌入式设备的管理与交互中，基于Web方式的应用成为目前的主流，这种程序结构也就是大家非常熟悉的B/S结构，即在嵌入式设备上运行一个支持脚本或CGI功能的Web服务器，能够生成动态页面，在用户端只需要通过Web浏览器就可以对嵌入式设备进行管理和监控，非常方便实用。本实验主要介绍这种应用的开发和移植工作。用户首先需要在嵌入式设备上成功移植支持脚本或CGI功能的Web服务器，如httpd，然后才能进行应用程序的开发。

(1)、嵌入式Web服务器的特点

嵌入式Web服务器和普通Web服务器一样，能够完成接收客户端请求、分析请求、响应请求、向客户端返回请求结果等任务。它的工作过程主要包括：

* 完成Web服务器的初始化工作，如创建环境变量、创建TCP套接字、绑定端口、开始侦听、进入循环结构，以及等待接收客户浏览器的连接请求；
* 当有客户端连接请求时，Web服务器负责接收客户端请求，并保存相关请求信息；
* 在接收到客户端的连接请求之后,分析客户端请求，解析出请求的方法、URL目标、可选的查询信息及表单信息，同时根据请求做出相应的处理；
* Web服务器完成相应处理后，向客户端浏览器发送响应信息，关闭与客户机的TCP连接。

(2)、嵌入式Web服务器移植

由于嵌入式设备资源一般都比较有限，并且也不需要能同时处理很多用户的请求，因此不会使用Linux下最常用的如Apache等服务器，而需要使用一些专门为嵌入式设备设计的Web服务器，这些Web服务器在存贮空间和运行时所占有的内存空间上都会非常适合于嵌入式应用场合。典型的嵌入式Web服务器有Boa（www.boa.org）和httpd等，它们和Apache等高性能的Web服务器主要的区别在于它们一般是单进程服务器，只有在完成一个用户请求后才能响应另一个用户的请求，而无法并发响应，但这在嵌入式设备的应用场合里已经足够了。

本实验采用busybox中的httpd作为实验的Web服务器。

(3)、CGI应用程序

CGI(Common Gateway Interface)是WWW技术中最重要的技术之一，有着不可替代的重要地位。CGI是外部应用程序（CGI程序）与WEB服务器之间的接口标准，是在CGI程序和Web服务器之间传递信息的过程。CGI规范允许Web服务器执行外部程序，并将它们的输出发送给Web浏览器，CGI将Web的一组简单的静态超媒体文档变成一个完整的新的交互式媒体。绝大多数的CGI程序被用来解释处理来自表单的输入信息，并在服务器产生相应的处理，或将相应的信息反馈给浏览器。CGI程序使网页具有交互功能。

CGI处理步骤：

* 通过Internet把用户请求送到web服务器。
* web服务器接收用户请求并交给CGI程序处理。
* CGI程序把处理结果传送给web服务器。
* web服务器把结果送回到用户。

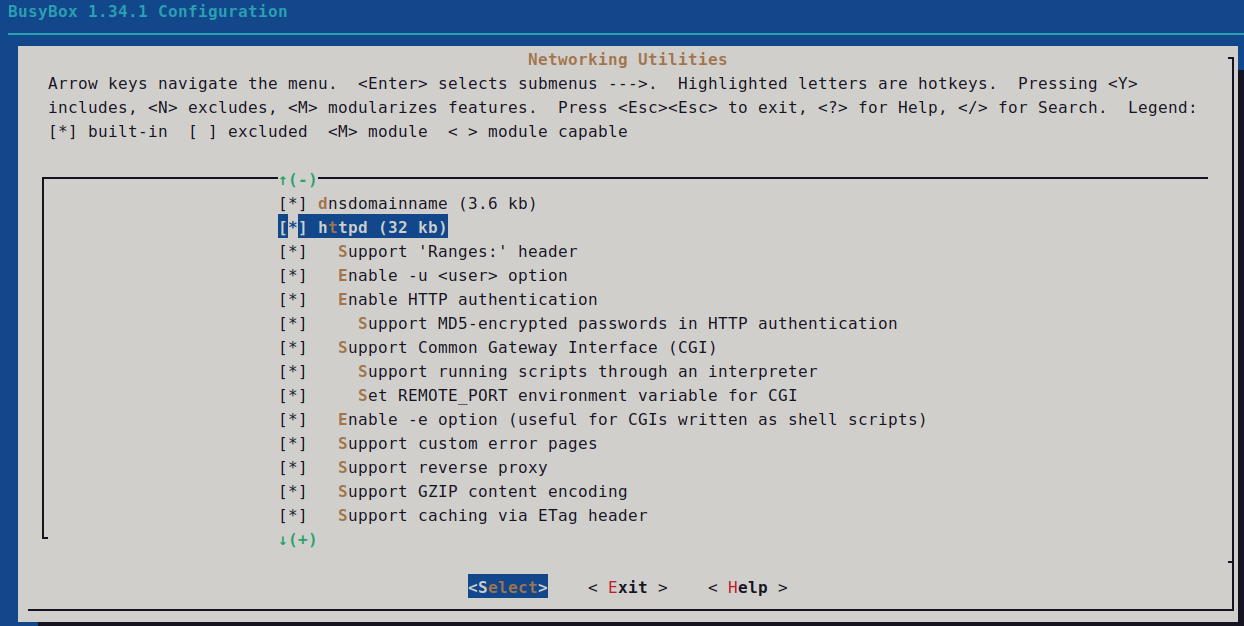
CGI可以用任何一种语言编写，只要这种语言具有标准输入、输出和环境变量。

CGI工作原理：

* 浏览器通过HTML表单或超链接请求指向一个CGI应用程序的URL。
* 服务器收发到请求。
* 服务器执行指定CGI应用程序。
* CGI应用程序执行所需要的操作，通常是基于浏览者输入的内容。
* CGI应用程序把结果格式化为网络服务器和浏览器能够理解的文档（通常是HTML网页）。
* 网络服务器把结果返回到浏览器中。

**5. 实验步骤**

（1）busybox目前默认配置中已经选择了httpd，大家需要在busybox编译前再次确认，如下图：



（2）httpd（WEB服务）测试

QEMU虚拟机图形化启动，在上面实验的armbin目录中建立启动脚本文件run3.sh，如下：

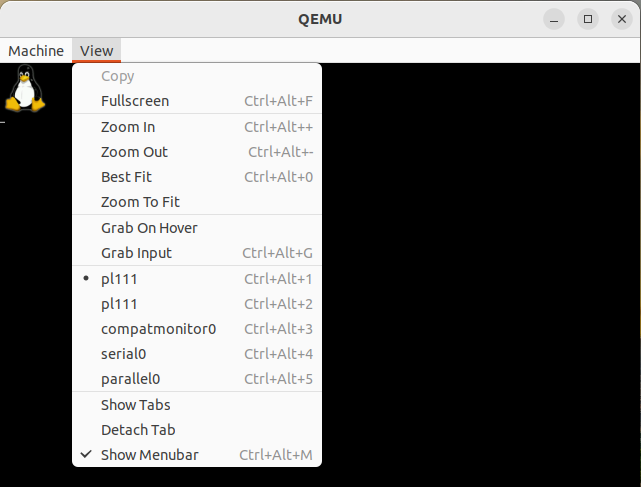
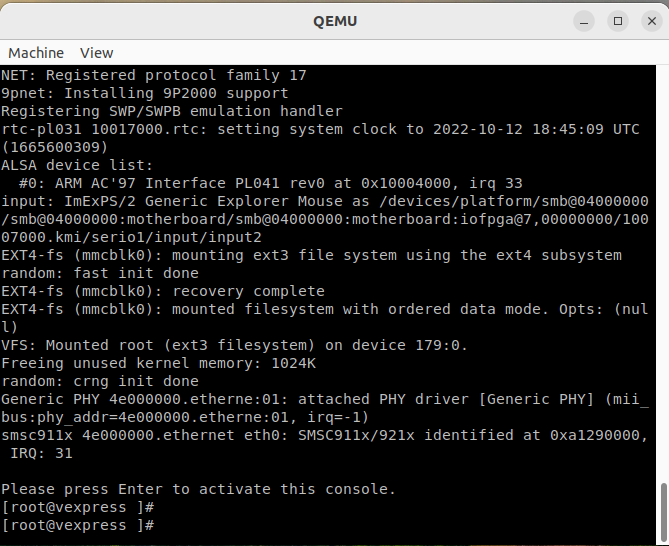
#run3.sh

qemu-system-arm -M vexpress-a9 -m 512M -dtb vexpress-v2p-ca9.dtb -kernel zImage -append "root=/dev/mmcblk0 rw console=ttyAMA0" -sd rootfs.ext3 -net nic -net user,hostfwd=tcp::8080-:8000 -display gtk,zoom-to-fit=true

添加run3.sh可执行权限：chmod +x run3.sh

QEMU虚拟机图形化启动：./run3.sh

QEMU虚拟机图形化启动后界面如下图，需要切换到serial0（串口）模式才能输入命令。

“pl111”菜单可以显示QEMU虚拟机的图形化屏幕，相当于LCD显示屏。

将老师提供的网页测试文件[www目录](http://www.tar.gz)拷贝到NFS共享目录中，然后修改网页文件index.html，将“测试者”改成自己的名字，在QEMU虚拟机中运行命令：

挂载NFS共享目录：mount -t nfs -o nolock 10.0.2.2:/home/zxf/nfs /mnt/nfs

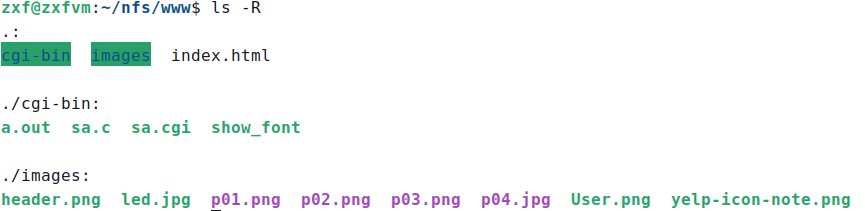
运行Web服务httpd：httpd -p 8000 -h /mnt/nfs

之后打开Ubuntu中的浏览器访问QEMU虚拟机中的WEB服务（浏览器中输入地址：http://192.168.254.142:8080），如下图效果：

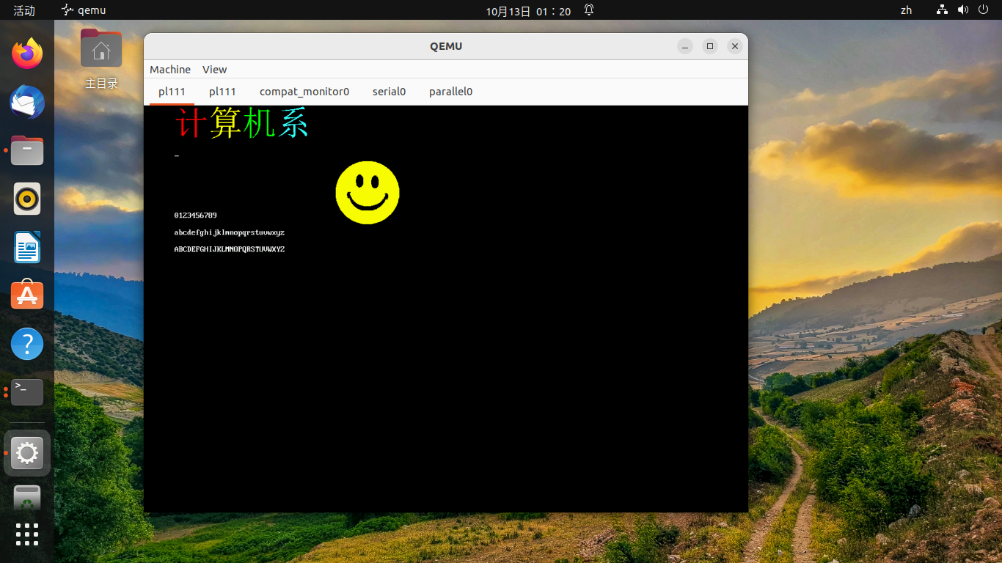


注意：“192.168.254.142”为Ubuntu本机IP地址，由于QEMU中已经设置了tcp转发功能，即访问“192.168.254.142”的8080端口会被转发到QEMU虚拟机中，则QEMU虚拟机中的httpd会响应，提供Web服务。

网站www目录结构如下图所示：

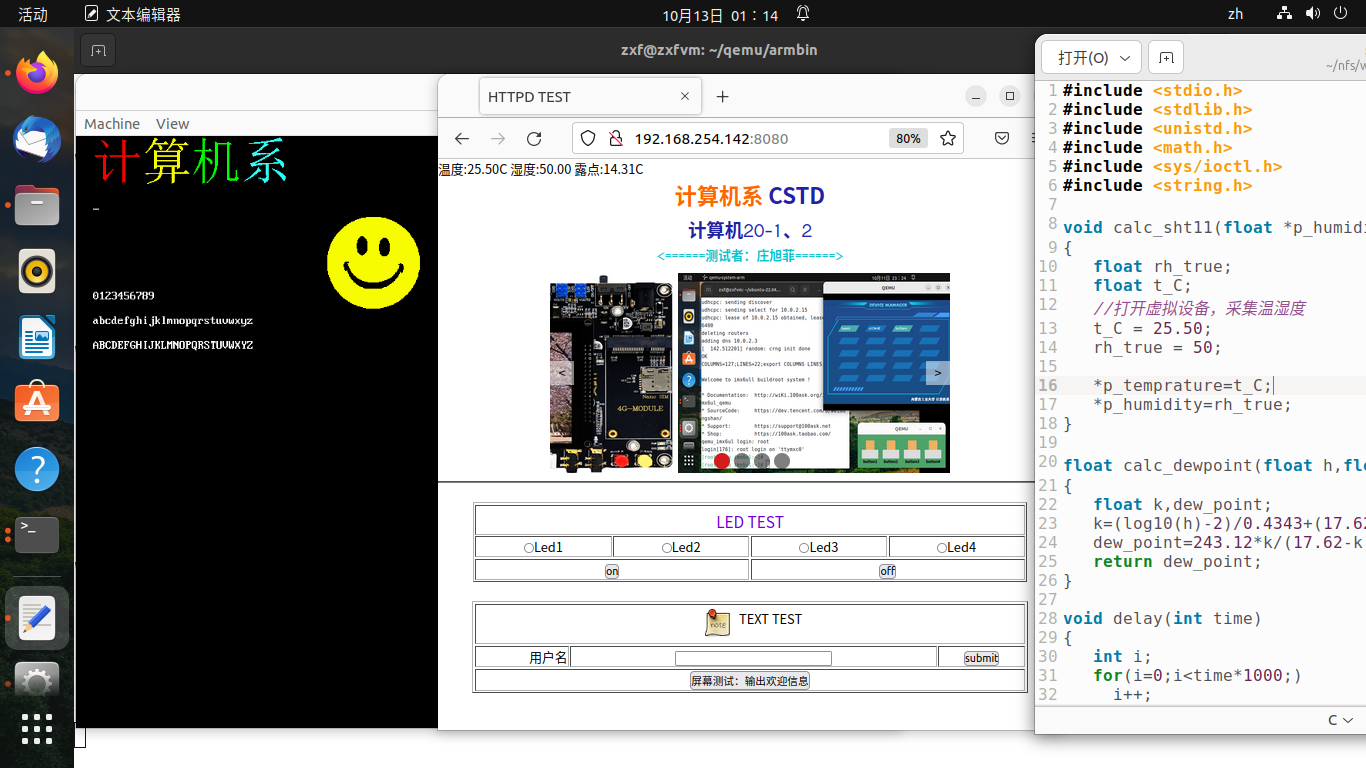


Index.html是网站主页；cgi-bin目录中是CGI程序，其中的sa.cgi是由sa.c交叉编译生成的可执行文件；show\_font是QEMU虚拟机的LCD测试程序，单独运行，如：./show\_font 可以显示如下图所示的界面（需要切换到“pl111”显示模式才能查看到）：

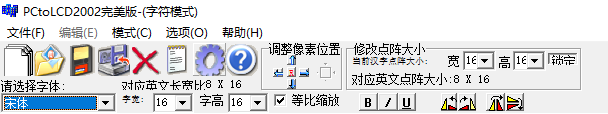
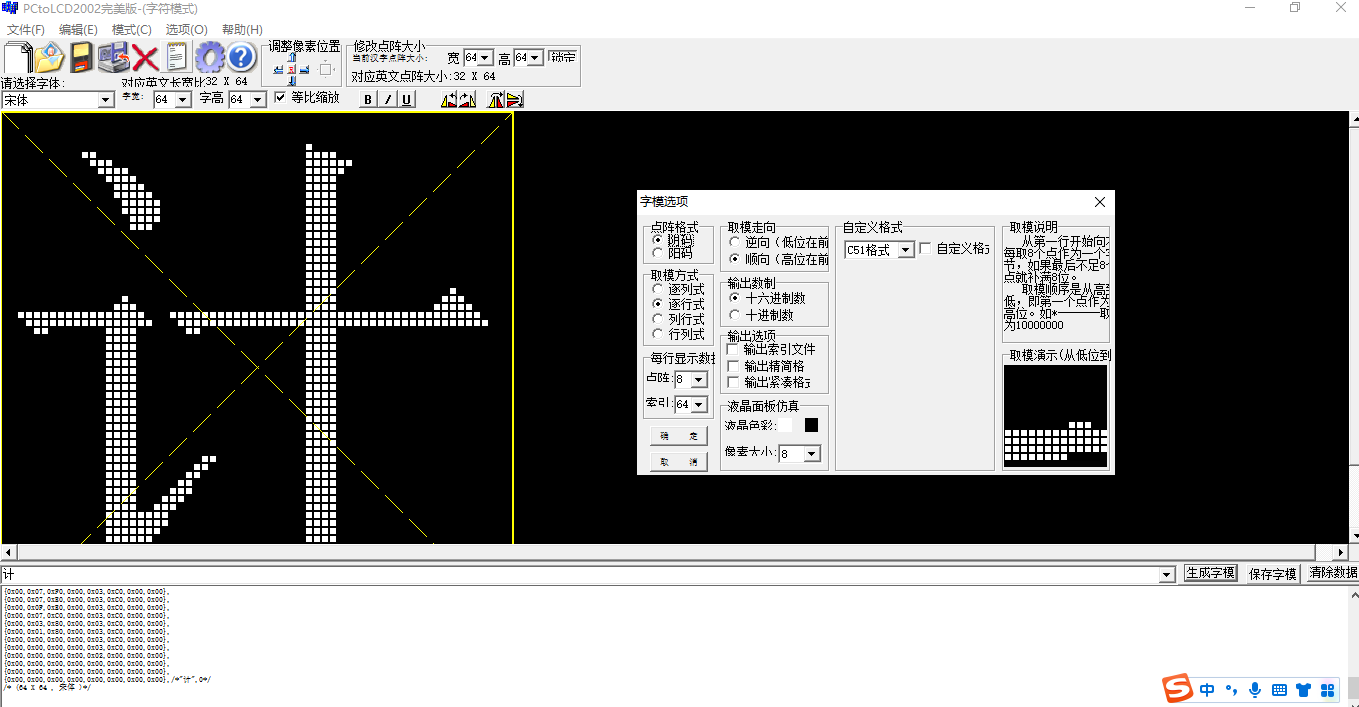


（13）请分析cgi-bin目录中的CGI程序sa.c的功能，要求修改sa.c，实现以下功能：

点击网页中的按钮可以在sa.c中调用实验三中的虚拟温湿度驱动程序，采集温湿度值并在网页左上角上显示出来；同时在QEMU虚拟机的LCD上显示如上图所示的LCD测试程序界面。最终效果如下图所示：



附1：QEMU虚拟机的LCD测试程序说明：

1. LCD测试源程序在实验资料的fbtest目录中。
2. show\_font.h中是点阵字库，show\_font.c是主程序。
3. 如果想要生成自己的字库，定制LCD输出的汉字，可以采用PCtoLCD2002软件生成汉字字模点阵，拷贝到show\_font.h中，并在show\_font.c中调用输出。
4. PCtoLCD2002软件设置及字模生成界面如下图所示：
5. 
6. 
7. 

****

****

**《嵌入式操作系统》实验报告**

**班级**

**学号**

**姓名**

**指导教师 庄旭菲**

**内蒙古工业大学**

**信息工程学院计算机系**

**2022年11月**

实验报告格式要求：

实验题目（如：实验一 Linux内核移植与编译实验）【黑体小三，居中】

一、实验目的

（五号字体）

二、实验内容

（实验主要步骤及实验结果）

【如果是验证性实验（如实验一、实验二），说明实验主要步骤及实验结果；如果是综合设计性实验（如实验三、四、五），还要说明程序结构或程序设计思路（可以画出程序流程图予以说明），给出程序清单及运行结果】

（五号字体）

三、实验中遇到的问题及解决方法

（五号字体）