**Linux初步**

**实验报告**

**班级： 种子1501**

**学号： U201513503**

**姓名：** **彭哲坤**

2018年9月9日

# 实验目的

• 熟练掌握Linux操作系统的使用

• 了解Linux操作系统的运作过程，理解内核与外围支撑系统的关系

• 通过实验定制Linux系统内核与外围支撑系统，加深对开源操作系统的认识

• 最终实现具有各自功能特色的自启动最小系统

# 实现步骤

整体要求：

1、bzImage < 4MB

2、initrd.img < 24MB

3、可通过console（主机界面）和ssh登录（ssh服务器）

4、支持ssh到其他主机（ssh客户端）

5、可挂载U盘

6、可访问windows NTFS分区（ntfs-3g）

7、可以正常关机（包括关闭电源）以及重启

外围支撑系统的实现步骤：

在获得shell版本的initrd.img基础上（v0.5）

– 完成拥有可以挂载原系统能力的v0.55

– 完成拥有管理设备能力（udev）的v0.6

– 完成拥有login登录能力（多窗口）的v0.7

– 达到由/sbin/init管理的小系统原型v0.9

– 实现网络功能、挂载U盘、访问硬盘分区功能的v1.0

之后分别实现内核定制与OS Loader安装。

# .实验内容

## 文件系统

### v0.5:shell版本的initrd.img

shell版本的Linux其实很简单。用一个/bin/bash和一个init脚本就可以。

init里写道：

#!/bin/bash

/bin/bash

其中第一行作为脚本解释器，从第二行才开始真正执行。

使用：

find . | cpio -H newc -o | gzip > /boot/initrd.img

将整个目录打包生成initrd.img，放/boot/下，并修改/boot/grub/grub.conf，增加：

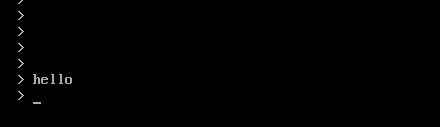
title CentOS test (V0.5)

root (hd0,0)

kernel /vmlinuz-2.6.32-696.el6.x86\_64 ro root=UUID=d493f959-fd0f-469d-980b-77b9f4262f60 rd\_NO\_LUKS rd\_NO\_LVM LANG=en\_US.UTF-8 rd\_NO\_MD SYSFONT=latarcyrheb-sun16 crashkernel=auto KEYBOARDTYPE=pc KEYTABLE=us rd\_NO\_DM quiet

initrd /inittmp.img

重启虚拟机后进入该系统，可得到如下界面：



这样的系统很简单，啥功能都没有，但是它给了我们一个最基础的框架，我们可以在这个基础之上进行功能添加。

### v0.55: 拥有可以挂载原系统能力

我最初的做法就是使用lsmod查看当前所有被加载的模块，再用ldd去把依赖库和模块自己拷过去，注意有些模块的依赖库本身也有依赖，需要再深入一层进行。之后在init脚本里将模块悉数insmod，即可使用。

有很多童鞋之后都很纠结如何减少不必要的modules以节约空间，咳咳，我这里讨了个巧，把之后用内核定制出来的模块作为最终需要的模块加入进来。当然，当时的我还是并不知道该怎么操作的，所以全部加进来也没有问题。

另外，modprobe，俗称模块探针，可以把模块连同它依赖的模块一同加载，可以避免模块之间因为加载先后要求而加载失败的现象。

另外还需要加载一些模块来存放这些模块

mount -t proc proc /proc >/dev/null 2>&1

mount -t sysfs sysfs /sys >/dev/null 2>&1

mount -t devtmpfs -omode=0755 devtmpfs /dev >/dev/null 2>&1

mkdir /dev/shm

mkdir /dev/pts

mount -t devpts -o gid=5,mode=620 devpts /dev/pts >/dev/null 2>&1

mount -t tmpfs tmpfs /dev/shm >/dev/null 2>&1

### v0.6: 拥有管理设备能力（udev）

这里可以直接使用udev挂载所有的模块，可以不用之前的insmod了，不过udev有很多依赖的东西，包括/etc/nsswitch.conf，其本身也有规则和配置文件（放在/lib/udev/和/etc/udev/）,总依赖大致如下：(有些并不是依赖，只是提前把这个功能加了进去)

/etc/udev/ /lib/udev/ /etc/init.d /etc/sysconfig etc/profile.d

/etc/nsswitch.conf /etc/mtab /etc/ld.so.cache /etc/localtime

/bin/ln /bin/awk /bin/cat /bin/chown /bin/ls

/bin/mknod /bin/mount /bin/umount /bin/vi /bin/mkdir

/bin/echo /bin/sh /bin/cp /bin/mv /bin/rm

/sbin/fdisk /sbin/lsmod /sbin/modprobe /sbin/rmmod /sbin/udevadm

/sbin/insmod /sbin/MAKEDEV /sbin/pidof /sbin/restorecon /sbin/start\_udev

/sbin/udevd /sbin/modinfo

虽然不用insmod了，但是需要添加一下语句来启动udev

udevd --daemon --resolve-names=never

udevadm trigger

### v0.7&0.9: 拥有login登录能力（多窗口）功能且由/sbin/init管理

login本身需要依赖pam机制以及security等，除此之外复杂的进程管理机制和名词（进程组、控制台等），因此最好用/sbin/init来启动。

恰好之前做完了操作系统，对这里理解更深了，这里使用exec族函数，可以让/sbin/init 以当前进程号启动，即/sbin/init取代/init执行动作。

执行顺序：udevd ——>fsck ——> remount——>service——>login prompt

——> mingetty + /bin/login

其中udevd ——>fsck ——> remount为第一部分，依赖/etc/rc.sysinit，第二部分service依赖/etc/rc，第三部分login prompt则直接执行/sbin/mingetty与/bin/login

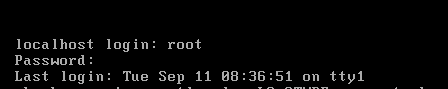
多窗口界面则可以在/etc/init/和/etc中寻找与tty有关的脚本进行配置。

pam机制。其主要依赖的是/etc/pam.d/、/lib64/security/、/etc/passwd、/etc/group、 /etc/shadow，悉数拷过去即可。如果想修改密码，可以先在本机将密码修改后将上述文件拷到initrd文件夹里即可

另外，关于能够正常开关机的操作，这里需要做的就是etc/rc.d/rc0.d 和etc/rc.d/rc6.d拷贝过去，因为实际上控制着运行级别的/etc/inittab，其内记录的运行级别x对应着执行rcx.d内的脚本，其中rc0.d对应着关机，rc6.d对应着重启。如果在开机关机的时候发现有什么问题（缺少目录、依赖等等），加上就好。



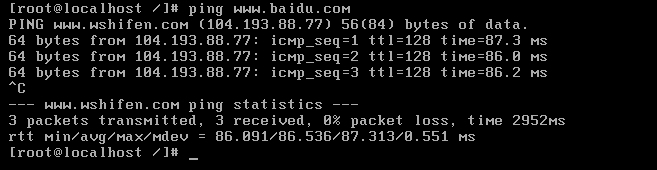
多用户登录：

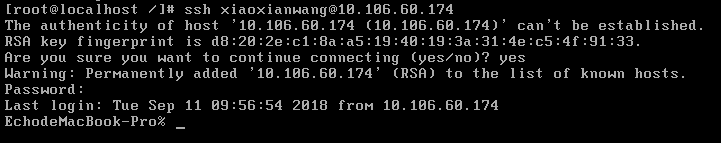
### v1.0 网络功能

网络功能所依赖的关键一点在于/etc/sysconfig/network-scripts/ifcfg-eth0中，配置好相应的IP，掩码，网关等，注意使用静态IP。

可上网：



可ssh(远程上晓纤的电脑)：



## ntfs-3g模块与U盘挂载

下载ntfs-3g\_ntfsprogs-2017.3.23.tgz, 使用tar解压，进入目录后使用：

./configure

Make

Make install

三连，并在/bin中找到编好的ntfs-3g，同样使用ldd找到依赖一并打包到new\_initrd即可。之后可以使用

ntfs-3g /dev/sda/ /mnt/disk/

完成挂载。

U盘挂载需要有三个模块，分别是usb-storage scsi\_mod sd\_mod，后面三个平时一直都在，但是usb-storage只有在插上U盘的时候才出现。如果忽略了，请再手动添加一次，这样才能被识别。

## 内核定制：

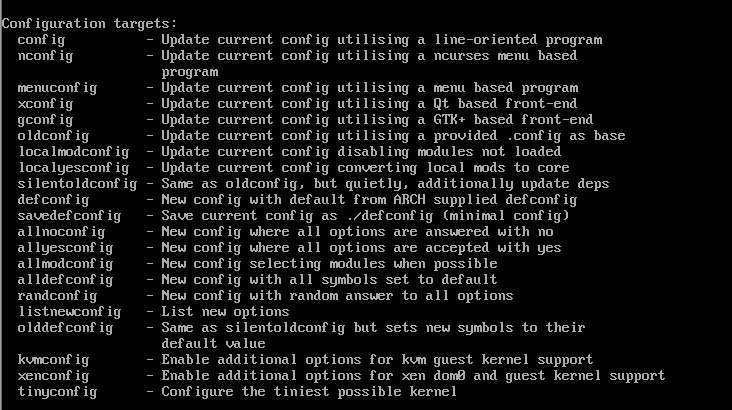
首先，从开源站上下载下来开源的源码包

wget <https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.13.9.tar.xz>

之后进行解压得到包内文件：



眼花缭乱，但是Makefile是那么得瞩目，在课上钟sir提过建议使用make menuconfig，但是之前我们先看一下make help，其中提供的选择：

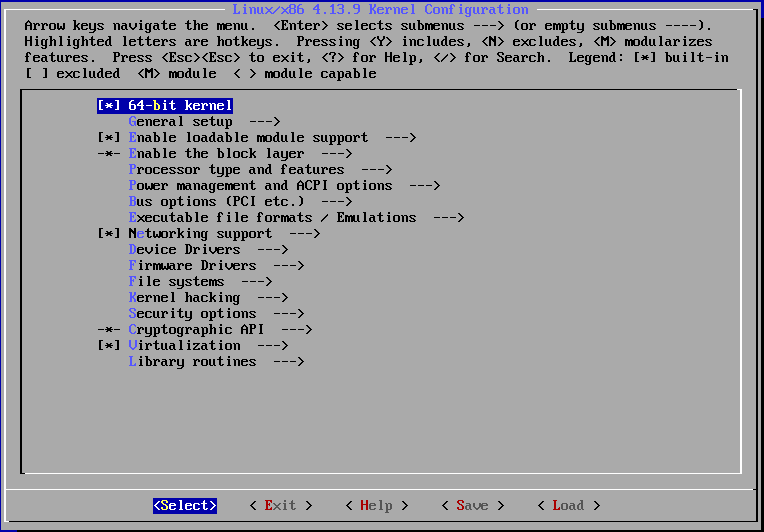


选项很多，但是只有前面的几个是我们可以用的，其中menuconfig的叙述是：Update current config utilizing a menu based program, 这样可以帮我们把一些必要的东西省略，所以直接进入！

make menuconfig

但是……需要一个ncurese-devel，用yum安装一下后继续。

出现menuconfig界面！



由于是64位系统，所以我选择了默认的64位kernel。

General setup



swap是Linux的交换分区，System V IPC是进程通信，另外请保持所有syscall，经过大量测试，syscall与其他某些未知模块存在耦合关系，如果没有完全去除，则会造成内核很慌。



需要其支持kernel config

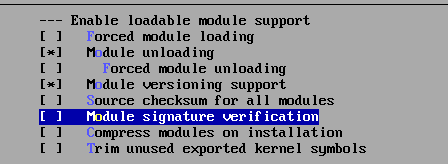


进程组调度和initrd支持都是必要的。



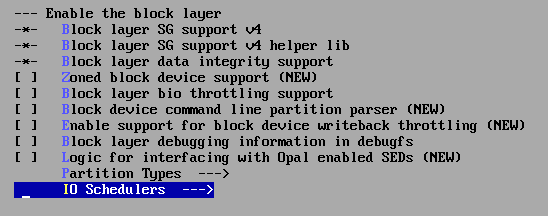
可调节大小

Enable loadable module support



这里忽略了所有其他的选项，只选择了模块去加载和版本支持（加载由文件系统提供）

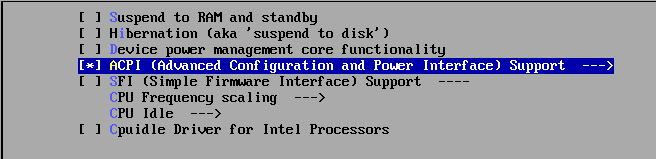
Enable Block Layer



Processor type and features

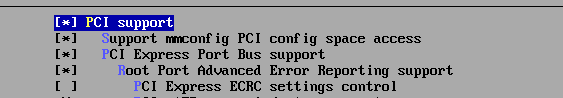
基本上除了Module其他的全部都不用，只是有一个Processor family（Generic-86-64）我选择的是64位系统的处理器

Power management and ACPI options



只保留ACPI

Bus options



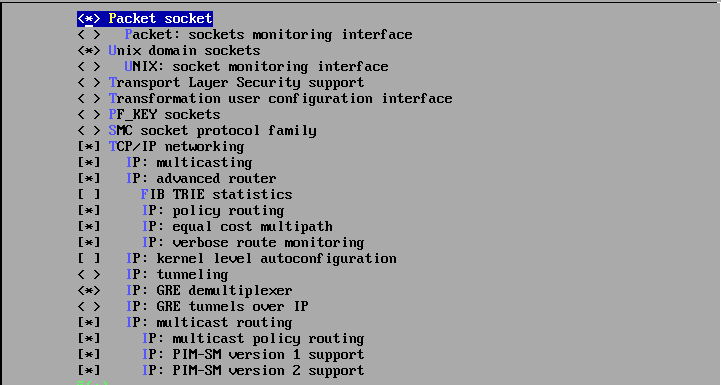
只选择PCI部分以及其下必要的支持

Executable file formats



ELF是一定要选的，其他的我保持原样，但是取消了core dump(这个debug信息对我们小系 统没有太大帮助)

Networking support



把Packet socket和Unix domain sockets留下，还把TCP/IP留下，其他的全部取消。

Device drivers

其他地方没做太大修改，但是把Ethernet driver support里的所有跟Intel和Realtek无关的东西全部取消了（我电脑上的网卡是这俩公司的产品，相信大部分人也都是）

File system

维持ext，其他的不变

make的时候死活动不起来，问题出在arch/x86/Makefile:175: \*\*\* Missing operator (did you mean tab instead of 8 spaces ), 但是打开看之后却发现不管怎么调tab还是空格都没有用。仔细看了一下上面的注释，说道：

# GCC ignores '-maccumulate-outgoing-args' when used with '-Os'.

# If '-Os' is enabled, disable it and print a warning.

ifdef CONFIG\_CC\_OPTIMIZE\_FOR\_SIZE

undefine CONFIG\_CC\_OPTIMIZE\_FOR\_SIZE

$(warning Disabling CONFIG\_CC\_OPTIMIZE\_FOR\_SIZE. Your compiler does not have -mfentry so you cannot optimize for size with CONFIG\_FUNCTION\_GRAPH\_TRACER.)

endif

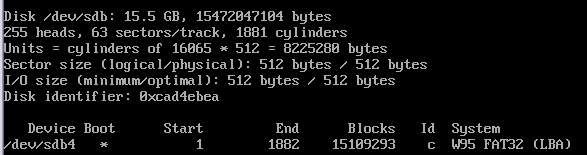
可见是因为gcc的-Os造成的，而这个-Os的选项是因为之前选择了Compiler Optimization level(Optimized by size), 这里的Makefile不想执行-Os，因为它会使得堆栈对齐失效，如果不disable也不会有太大问题。但是按理说下面这几句执行的语句也没有错啊。而且这个关乎着编译出来的大小，所以我无法绕过。于是我走上了一条不归路。升级GCC……长达4个小时，从4.4.7升级到6.2.0.再次尝试后成功！

另外，如果想压得小一点，建议选择好一点的压缩方式~

之后进一步筛选的时候，除了必要的硬件设备驱动和功能，其他的诸如Graphic等都被删得一干二净以节约空间。

## OS Loader – Grub

之后用fdisk对U盘进行分区和格式化，



这里我将U盘的第四个分区进行格式化成启动分区，用mkfs.vfat将其格式化成fat格式

之后以vfat格式挂载：

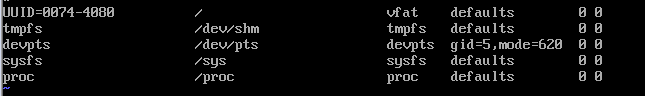
mount–t vfat /dev/sdb4 /mnt/usb

先将grub装入系统，使用

grub-install –root-directory=/mnt/usb /dev/sdb4

然后按照U盘的UUID修改/etc/fstab，从而在文件系统启动时将其挂载到根目录下。

fstab：



注意将fstab修改后再将整个系统打包一次。

之后将原系统的/boot/grub/下所有文件也移进U盘中的同级目录。开始在U盘挂载的目录下安装引导，这里安装MBR引导，先进入grub，用find /boot/grub/stage1查找根目录，然后root (hdx,0), 再setup (hdx)，完成之后记得把打包后的内核和img一同拷到U盘的/boot中。注意把和内核配套的modules也拷到img内。之后通过修改电脑的启动顺序（我的电脑是Legacy First，把U盘启动优先级上调即可），进入后就可以开始体验小系统了！

# 调试过程

1. 首先最开始的时候，一个简单的init都没有办法起起来，最后居然逼得自己用C语言写了一个程序，用exec启动了bash才进入了小系统。

其原因在于：没有chmod +x，而C语言编译完成后自己就有了执行的权限。

1. 缺少录屏软件，但是在虚拟机中是可以用shift+PgUp和PgDn来看之前的信息的
2. 内核裁剪的时候裁得太high了，一次性减掉好多东西，之后kernel panic的时候也没法子确定多删了哪个模块，只能从头编译。而且.config文件没有保存，结果就是最终需要重新筛选。
3. make的时候经常出错，有的时候不要找偏了，因为编译器会很明显地告诉我们哪里出错，根据提示一个一个改好就行。稳定版本的错误相对较少。

# 分析总结

结果：bzImage:2.2M

initrd.img:23M

整个Linux系统还是很庞大的，但是我们只需要其中的最关键所在。这也就要求我们在探索的时候尽可能减少对无关事件的关注，尽可能把经历投入在主要的部分。

Linux初步里还是有很多的坑点的，公共的坑点就是源码包的版本问题，很多版本的源码都存在着或多或少的问题，有时候就是考验学习者的耐心，尽力去一点一点排查错误。并且读源码，裁内核，都需要仔细去分析关键问题所在，尽可能实现相应的功能，减少冗余。

自己的拖延症还是有点严重，不到最后关头不投入精力，也没有做到尽可能的完善。Linux考验的是长期积累，短时间内是没有办法做到深入了解的。是“初步”，却也很深入。