



中山大學
SUN YAT-SEN UNIVERSITY

本科生实验报告

实验课程: 操作系统原理实验

任课教师: 刘宁

实验题目: 第零章 编译内核/利用已有内核构建 OS

专业名称: 计算机科学与技术

学生姓名: 郑鸿鑫

学生学号: 22336313

实验地点: 实验中心 D503

实验时间: 2024/2/26

Section 1 实验概述

本次实验中，主要目的是熟悉现有 Linux 内核的编译过程和启动过程，并在自行编译内核的基础上构建简单应用并启动。同时利用精简的 Busybox 工具集构建简单的 OS，熟悉现代操作系统的构建过程。此外，也可以熟悉编译环境、相关工具集，并能够实现内核远程调试。

Section 2 预备知识与实验环境

- 预备知识：x86 汇编语言程序设计、Linux 系统命令行工具
- 实验环境：
 - 虚拟机版本/处理器型号：
11th Gen Intel® Core™ i5-11320H @ 3.20GHz × 2
 - 代码编辑环境：VS Code
 - 代码编译工具：g++
 - 重要三方库信息：Linux 内核版本号：linux-5.10.210
Ubuntu 版本号：Ubuntu 18.04.6LTS，Busybox 版本号：
Busybox_1_33_0

Section 3 实验任务

- 实验任务 1：搭建 OS 内核开发环境包括：代码编辑环境、编译环境、运行环境、调试环境等。
- 实验任务 2：下载并编译 i386（32 位）内核，并利用 qemu 启动内核。
- 实验任务 3：熟悉制作 initramfs 的方法，编写简单应用程序随内核启动运行，开启远程调试功能，进行调试跟踪代码运行。
- 实验任务 4：编译 i386 版本的 Busybox，随内核启动，构建简单的 OS。

Section 4 实验步骤与实验结果

----- 实验任务 1 -----

- 任务要求：搭建 OS 内核开发环境。
- 思路分析：依照指导书上指引配置环境。

- 实验步骤:

1. 在清华下载源安装 Ubuntu
2. 然后使用 gedit 打开下载源保存的文件/etc/apt/sources.list
3. 将下载源复制进/etc/apt/sources.list 后保存退出。

配置 C++环境, 在终端输入以下代码:

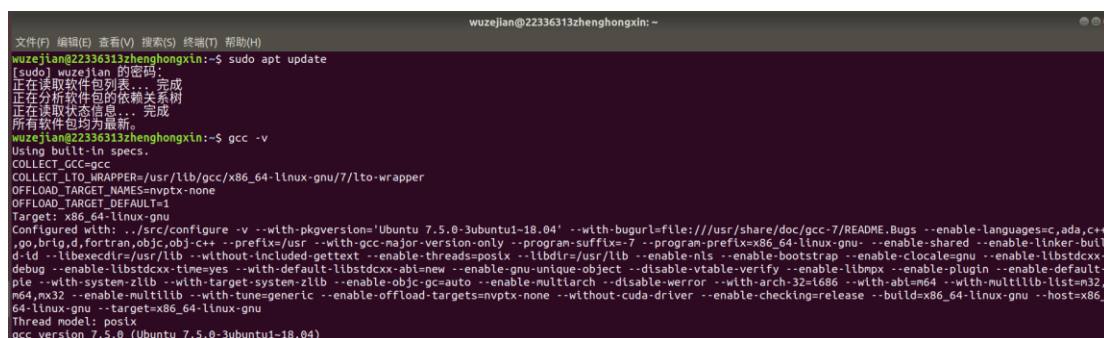
```
sudo apt install binutils  
sudo apt install gcc
```

4. 安装 VSCode 和其他工具:

```
sudo apt install qemu  
sudo apt install cmake  
sudo apt install libncurses5-dev  
sudo apt install bison  
sudo apt install flex  
sudo apt install libssl-dev  
sudo apt install libc6-dev-i386  
sudo apt install gcc-multilib  
sudo apt install g++-multilib
```

- 实验结果展示:

如下图所示: 已更换为清华下载源且 gcc 版本号为 7.5.0



```
wuzejian@22336313zhenghongxin:~$ sudo apt update  
[sudo] wuzejian 的密码:  
正在读取软件包列表... 完成  
正在分析软件包的依赖关系树  
正在读取状态信息... 完成  
所有软件包均都是最新的。  
wuzejian@22336313zhenghongxin:~$ gcc -v  
Using built-in specs.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper  
OFFLOAD_TARGET_NAMES=nvptx-none  
OFFLOAD_TARGET_DEFAULT=1  
Target: x86_64-linux-gnu  
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1-18.04' --with-bugurl=file:///usr/share/doc/gcc-7/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++ --prefix=/usr --with-gcc-major-version-only --program-suffix=-7 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-bootstrap --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib --enable-objc-gc=auto --enable-multitarget --disable-werror --with-arch=32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu  
Thread model: posix  
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1-18.04)
```

----- 实验任务 2 -----

- 任务要求: 下载并编译 i386 (32 位) 内核, 并利用 qemu 启动内核。
- 思路分析: 依照参考书指引完成编译和启动内核。
- 实验步骤:

1. 在用户目录下创建文件夹 lab1 并进入

```
mkdir ~/lab1  
cd ~/lab1
```

2. 到 <https://www.kernel.org/> 下载内核 5.10.210 到文件夹~/lab1。

3. 解压并进入。

```
xz -d linux-5.10.19.tar.xz  
tar -xvf linux-5.10.19.tar  
cd linux-5.10.19
```

5. 将内核编译为 i386 32 位版本：

```
make i386_defconfig  
make menuconfig
```

6. 编译内核：

```
make -j8
```

7. 使用 qemu 启动内核：

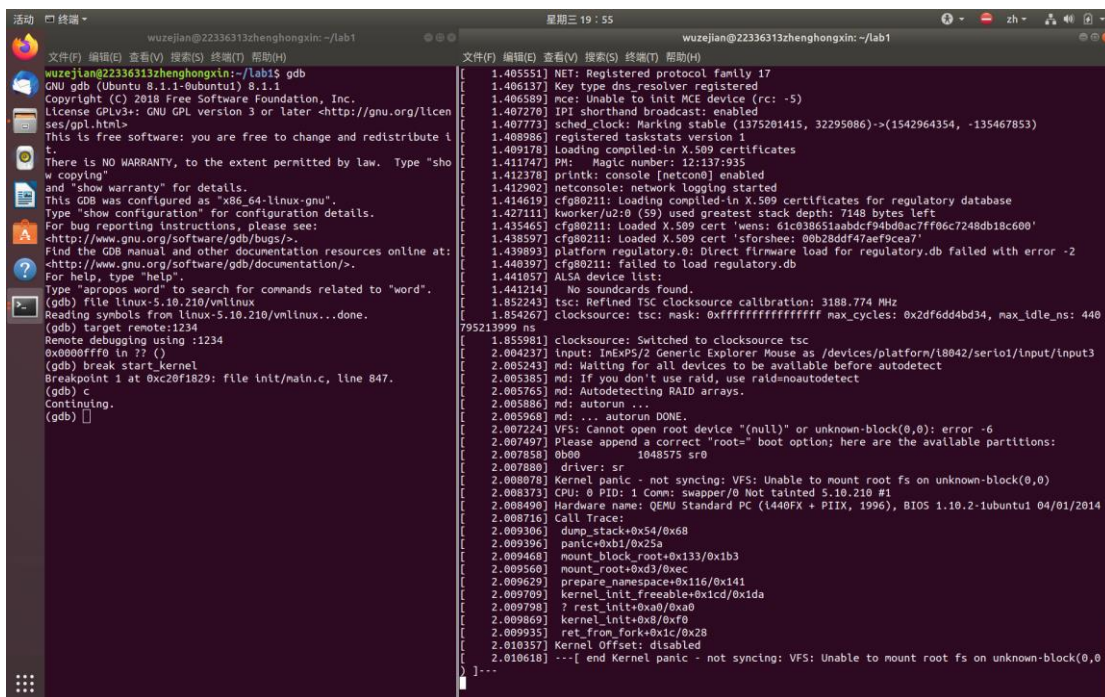
```
qemu-system-i386 -kernel linux-5.10.210/arch/x86/boot/bzImage -s -S -  
append "console=ttyS0" -nographic
```

8. 启动 gdb 调试，在另一个终端下启动 gdb，且不关闭 qemu 所在的终端，加载符号表和链接已启动的 qemu 进行调试，在 gdb 下为 start_kernel 函数设置断点，最后输入 c 运行

```
gdb  
file linux-5.10.210/vmlinux  
target remote:1234  
break start_kernel  
c
```

● 实验结果展示：

由下图可知 linux 压缩镜像 bzImage 已生成



实验任务 3

- 任务要求: 熟悉制作 initramfs 的方法, 编写简单应用程序随内核启动运行, 开启远程调试功能, 进行调试跟踪代码运行。
- 思路分析: 依照指导书的指引完成 initramfs 的制作。
- 实验步骤:

1. 制作一个简单的 Hello World initramfs, 如下:

```
#include <stdio.h>

void main()
{
    printf("lab1: Hello World\n");
    fflush(stdout);

    /* 让程序打印完后继续维持在用户态 */
    while(1);
}
```

2. 将上述代码编译为 32 位可执行文件

```
gcc -o helloworld -m32 -static helloworld.c
```

3. 用 cpio 打包 initramfs:

```
echo helloworld | cpio -o --format=newc > hwinitramfs
```

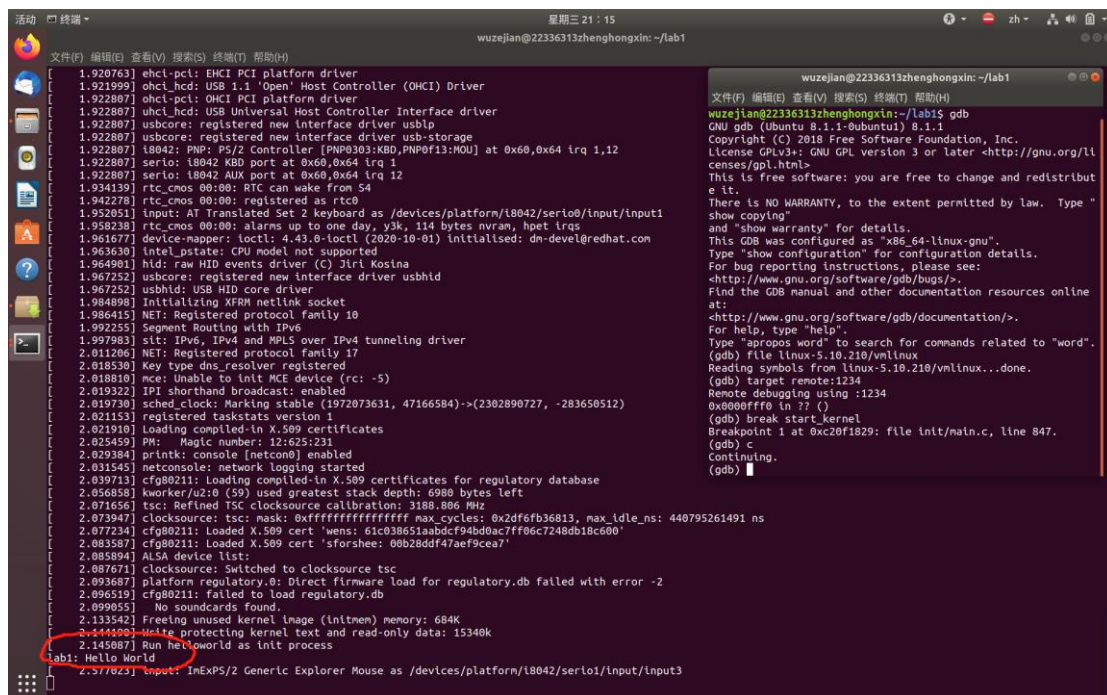
4. 启动内核，并加载 initramfs:

```
qemu-system-i386 -kernel linux-5.10.210/arch/x86/boot/bzImage -initrd  
hwinitramfs -s -S -append "console=ttyS0 rdinit=helloworld" -nographic
```

5. 重复上述 gdb 调试过程。

- 实验结果展示:

由下图可知，gdb 中输出了 Hello World\n



实验任务 4

- 任务要求: 编译 i386 版本的 Busybox, 随内核启动, 构建简单的 OS。
- 思路分析: 依照指导书的指引完成 busybox 的下载和编译及启动。
- 实验步骤:

1. 从网站下载 Busybox 到 lab1, 然后解压:

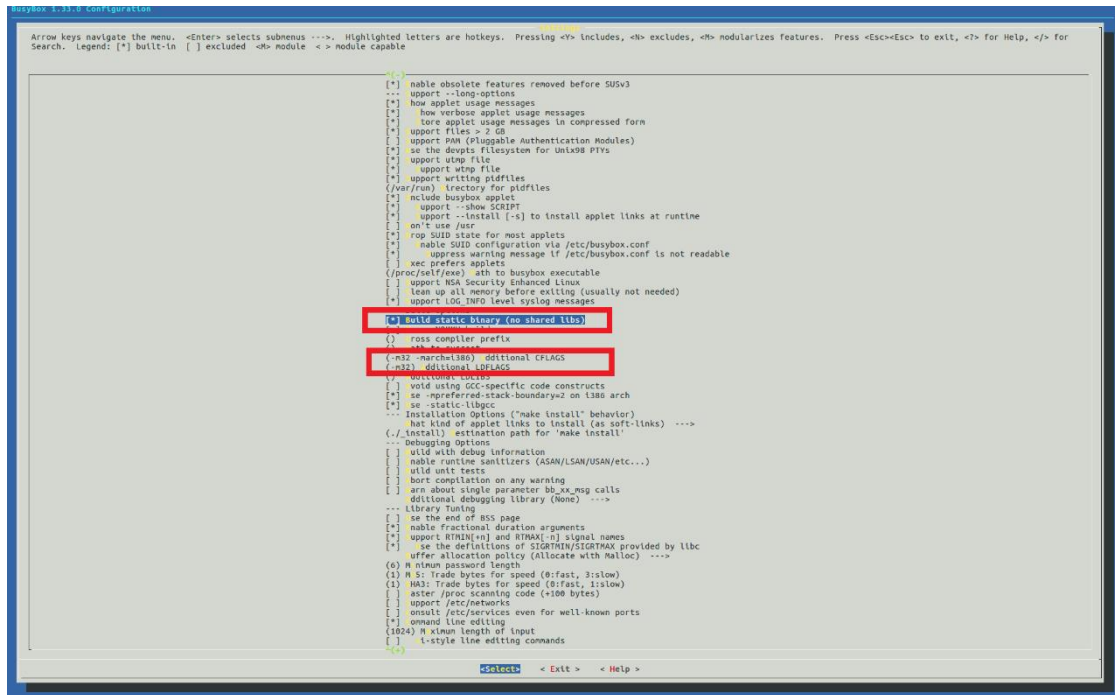
```
tar -xf Busybox_1_33_0.tar.gz
```

2. 编译 Busybox:

```
make defconfig  
make menuconfig
```

3. 进入 settings, 然后在 Build BusyBox as a static binary(no shared libs) 处输入 Y 勾选, 然后分别设置 () Additional CFLAGS 和 () Additional LDFLAGS 为(-m32 -march=i386) Additional CFLAGS 和(-

m32) Additional LDFLAGS:



4. 保存退出后编译:

```
make -j8
```

```
make install
```

5. 将安装在_install 目录下的文件和目录都取出复制到新建的一个 mybusybox 文件夹中:

```
cd ~/lab1
```

```
mkdir mybusybox
```

```
mkdir -pv mybusybox/{bin,sbin,etc,proc,sys,usr/{bin,sbin}}
```

```
cp -av busybox-1_33_0/_install/* mybusybox/
```

```
cd mybusybox
```

6. 写一个简单的 shell 脚本作为 init。用 gedit 打开文件 init，复制入如下内容，保存退出:

```
#!/bin/sh
```

```
mount -t proc none /proc
```

```
mount -t sysfs none /sys
```

```
echo -e "\nBoot took $(cut -d' ' -f1 /proc/uptime) seconds\n"
```

```
exec /bin/sh
```


注意到第一句标红的指定解释器在指导书中没有，而是参考参考文档后添加上，解决了后续加载 busybox 时出现的无法打开文件夹的问题。

7. 加上执行权限：

```
chmod u+x init
```

8. 将 x86-busybox 打包归档位 cpio 文件，以供 Linux 内核做 initramfs 启动执行：

```
find . -print0 | cpio --null -ov --format=newc | gzip -9 >
~/lab1/initramfs-busybox-x86.cpio.gz
```

9. 加载 busybox：

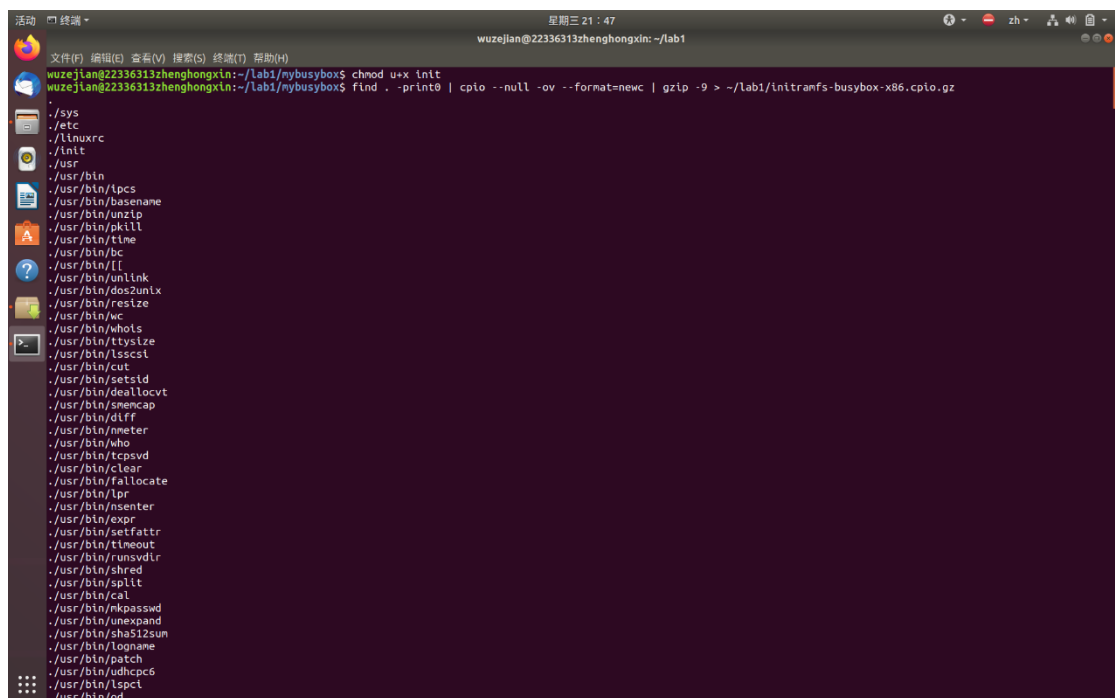
```
cd ~/lab1

qemu-system-i386 -kernel linux-5.10.210/arch/x86/boot/bzImage -
initrd initramfs-busybox-x86.cpio.gz -nographic -append
"console=ttyS0"
```

然后用 `ls` 命令可以看到当前文件夹。

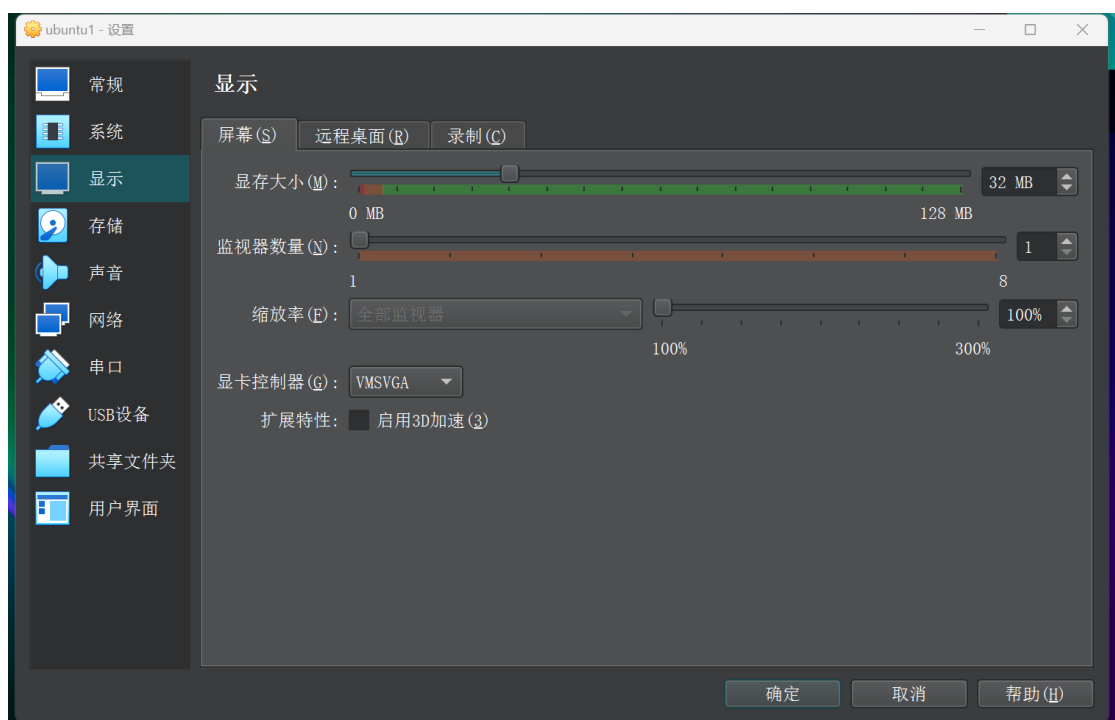
● 实验结果展示：

如下图所示，可以看到 busybox 成功加载并随内核启动，在最后一张图片中可以看到文件夹中的文件：




```
wuzejian@ubuntu1: ~/lab1/linux-5.10.210
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
您希望继续执行吗? [Y/n] Y
获取:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic/universe amd64 make-guile amd64 4.1-9.1ubuntu1 [155 kB]
已下载 155 kB, 耗时 0秒 (320 kB/s)
(正在读取数据库 ... 系统当前共安装有 133051 个文件和目录。)
正在卸载 make (4.1-9.1ubuntu1) ...
正在选中未选择的软件包 make-guile。
(正在读取数据库 ... 系统当前共安装有 133036 个文件和目录。)
正准备解包 .../make-guile_4.1-9.1ubuntu1_amd64.deb ...
正在解包 make-guile (4.1-9.1ubuntu1) ...
正在设置 make-guile (4.1-9.1ubuntu1) ...
正在处理用于 man-db (2.8.3-2ubuntu0.1) 的触发器 ...
wuzejian@ubuntu1:~/lab1/linux-5.10.210$ make i386_defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX scripts/kconfig/lexer.lex.c
/bin/sh: 1: flex: not found
scripts/Makefile.host:9: recipe for target 'scripts/kconfig/lexer.lex.c' failed
make[1]: *** [scripts/kconfig/lexer.lex.c] Error 127
Makefile:615: recipe for target 'i386_defconfig' failed
make: *** [i386_defconfig] Error 2
wuzejian@ubuntu1:~/lab1/linux-5.10.210$
```

2. 在 Ubuntu 中安装增强功能后，会出现黑屏，本人一开始认为是操作哪里出错，删除虚拟机重试后仍会黑屏，经过上网搜索发现，需要调整虚拟机的显存：（改为 32MB 或以上即可）



3. 在编译过程用到的 init 脚本，应该是新建在 mybusybox 文件夹中而不是 lab1 中，本人一开始将其放在 lab1 中导致后续打包过程过久，打包了 700 多万块，后通过理解指导书上下文后改正。

体会与收获:

通过此次实验，对 Linux 的内核的编译和启动过程有了一定的了解，并学会了利用 Busybox 构建简单的 OS，实现内核远程调试。对 Linux 中的命令也有了更多的了解。

Section 6 对实验的改进建议和意见

在加载 busybox 过程中，最后通过 ls 命令查看文件夹时会出现以下情况：

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
```

```
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty3: No such file or directory
```

经过 QQ 群里同学们的交流和参考老师给的实验参考文档后，在 `init` 脚本中增加一句 `#!/bin/sh` 作为指定解释器可以解决该问题，其中指导书中并未给出。（详见上文 Section4 实验任务 4 的第 6 步）

Section 7 附录：参考资料清单

指导书网站: [SYSU-2023-Spring-Operating-System: 中山大学 2023 学年春季操作系统课程 - Gitee.com](#)

参考文档: QEMU 上运行 BusyBox 详解.pdf



你好, 游客 登录 注册



发现更多精彩
扫码关注Linux公社微信公众号



[首页](#) [Linux新闻](#) [Linux教程](#) [数据库技术](#) [Linux编程](#) [服务器应用](#) [Linux安全](#) [Linux下载](#) [Linux主题](#) [Linux壁纸](#) [Linux软件](#) [数码](#) [手机](#) [电脑](#)

[首页](#) → [Linux教程](#)

阅读新闻	背景: □□□□□□□□	最	
<div>QEMU上运行BusyBox详解</div> <div>[日期: 2019-03-30]</div> <div>来源: Linux社区 作者: 醉落红尘</div> <div>[字体: 大 中 小]</div> <div>BusyBox</div> <div>前文"在QEMU环境中使用GDB测试Linux内核"和"Initramfs 原理和实践" 分别描述了怎么用qemu来运行一个编译好的内核, 以及怎么指定initramfs, 但都是简单的演示。其实轮子已经有人造出来了, BusyBox项目就是这样 一个工具集, 提供了非常多的常用Linux命令, 并且支持多平台。BusyBox项目的官网介绍如下:</div> <div>BusyBox: The Swiss Army Knife of Embedded Linux</div> <div>BusyBox combines tiny versions of many common UNIX utilities into a single small executable. It provides replacements for most of the utilities you usually find in GNU fileutils, shellutils, etc. The utilities in BusyBox generally have fewer options than their full-featured GNU cousins; however, the options that are included provide the expected functionality and behave very much like their GNU counterparts. BusyBox provides a fairly complete</div>			<div>正</div> <div>M</div> <div>M</div> <div>A</div> <div>P</div> <div>P</div> <div>P</div> <div>P</div> <div>J</div> <div>P</div>