《操作系统综合实验》

实验报告

07 Linux内核编译

姓 名　王 栗 政
班 级　2018039
学 号　20179100018
教 师　冯鹏斌

实验日期　2022.5.28

一、实验题目

```
下载、编译内核源代码

启动测试所编译出来的内核
```

二、相关原理与知识

1. 查看Linux内核版本

```
Usage: uname [OPTION]...
Print certain system information.  With no OPTION, same as -s.

  -a, --all                print all information, in the following order,
                             except omit -p and -i if unknown:
  -s, --kernel-name        print the kernel name
  -n, --nodename           print the network node hostname
  -r, --kernel-release     print the kernel release
  -v, --kernel-version     print the kernel version
  -m, --machine            print the machine hardware name
  -p, --processor          print the processor type (non-portable)
  -i, --hardware-platform  print the hardware platform (non-portable)
  -o, --operating-system   print the operating system
      --help     display this help and exit
      --version  output version information and exit
```

2. Linux内核文件

```
几种linux内核文件的区别:

1、vmlinux   编译出来的最原始的内核文件，未压缩。
2、zImage    是vmlinux经过gzip压缩后的文件。
3、bzImage bz表示“big zImage”，不是用bzip2压缩的。两者的不同之处在于，zImage解压缩内核到低端内存(第一个640K)，bzImage解压缩内核到高端内存(1M以上)。如果内核比较小，那么采用zImage或bzImage都行，如果比较大应该用bzImage。
4、uImage    U-boot专用的映像文件，它是在zImage之前加上一个长度为0x40的tag。
5、vmlinuz   是bzImage/zImage文件的拷贝或指向bzImage/zImage的链接。
6、initrd    是“initial ramdisk”的简写。一般被用来临时的引导硬件到实际内核vmlinuz能够接管并继续引导的状态。
```

3. Linux下文件打包、解压缩指令

```
    *.Z            //     compress程序压缩产生的文件(现在很少使用)
```

```
    *.gz            //    gzip程序压缩产生的文件
    *.bz2           //    bzip2程序压缩产生的文件
    *.zip          //    zip压缩文件
    *.rar          //    rar压缩文件
    *.7z            //    7-zip压缩文件
    *.tar           //    tar程序打包产生的文件
    *.tar.gz        //    由tar程序打包并由gzip程序压缩产生的文件
    *.tar.bz2       //    由tar程序打包并由bzip2程序压缩产生的文件
```

gzip:

gzip可以压缩产生后缀为 .gz 的压缩文件，也可以用于解压gzip、compress等程序压缩产生的文件。不带任何选项和参数使用gzip或只带有参数 - 时，gzip从标准输入读取输入，并在标准输出输出压缩结果。

基础格式：gzip [Options] file1 file2 file3

指令选项：(默认功能为压缩)

```
-c              //将输出写至标准输出，并保持原文件不变
-d              //进行解压操作
-v              //输出压缩/解压的文件名和压缩比等信息
-digit          //digit部分为数字(1-9)，代表压缩速度，digit越小，则压缩速度越快，但压缩效果越差，digit越大，则压缩速度越慢，压缩效果越好。默认为6.
```

bzip2:

bzip2是采用更好压缩算法的压缩程序，一般可以提供较之gzip更好的压缩效果。其具有与gzip相似的指令选项，压缩产生 .bz2 后缀的压缩文件。

基础格式：bzip2 [Options] file1 file2 file3

指令选项：(默认功能为压缩)

```
-c              //将输出写至标准输出
-d              //进行解压操作
-v              //输出压缩/解压的文件名和压缩比等信息
-k              //在压缩/解压过程中保留原文件
-digit          //digit部分为数字(1-9)，代表压缩速度，digit越小，则压缩速度越快，但压缩效果越差，digit越大，则压缩速度越慢，压缩效果越好。默认为6.
```

打包指令——tar:

gzip 或 bzip2 带有多个文件作为参数时，执行的操作是将各个文件独立压缩，而不是将其放在一起进行压缩。这样就无法产生类似于Windows环境下的文件夹打包压缩的效果。(gzip与bzip2也可以使用文件夹作为参数，使用 -f 选项，但也是将其中的每个文件独立压缩)。为了实现打包压缩的效果，可以使用命令 tar 进行文件的打包操作(archive)，再进行压缩。

基本格式：tar [Options] file_archive //注意tar的第一参数必须为命令选项，即不能直接接待处理文件

常用命令参数：

```
//指定tar进行的操作，以下三个选项不能出现在同一条命令中
-c              //小写,创建一个新的打包文件(archive)
-x              //对打包文件(archive)进行解压操作
-t              //查看打包文件(archive)的内容,主要是构成打包文件(archive)的文件名

//指定支持的压缩/解压方式，操作取决于前面的参数，若为创建(-c),则进行压缩，若为解压(-x),则进行解压，不加下列参数时，则为单纯的打包操作(而不进行压缩)，产生的后缀文件为.tar
-z              //使用gzip进行压缩/解压，一般使用.tar.gz后缀
-j              //使用bzip2进行压缩/解压，一般使用.tar.bz2后缀
```

```
    //指定tar指令使用的文件，若没有压缩操作，则以.tar作为后缀
    -f filename     //-f后面接操作使用的文件，用空格隔开，且中间不能有其他参数，推荐放在参数集最后或单独作为参数
                    //文件作用取决于前面的参数，若为创建(-c),则-f后为创建的文件的名字(路径)，若为(-x/t),则-f后为待解压/查看的打包压缩文件名

    //其他辅助选项
    -v              //详细显示正在处理的文件名
    -C Dir          //大写,将解压文件放置在 -C 指定的目录下
    -p(小写)        //保留文件的权限和属性，在备份文件时较有用
    -P(大写)        //保留原文件的绝对路径，即不会拿掉文件路径开始的根目录，则在还原时会覆盖对应路径上的内容
    --exclude=file //排除不进行打包的文件

xz:
    解压tar.xz文件：先 xz -d xxx.tar.xz 将 xxx.tar.xz解压成 xxx.tar 然后，再用 tar xvf xxx.tar来解包。
```

4. `qemu` 相关知识

```
QEMU是一种通用的开源计算机仿真器和虚拟器。QEMU共有两种操作模式
    全系统仿真：能够在任意支持的架构上为任何机器运行一个完整的操作系统
    用户模式仿真：能够在任意支持的架构上为另一个Linux/BSD运行程序
同时当进行虚拟化时，QEMU也可以以接近本机的性能运行KVM或者Xen。

安装：
$ sudo apt install qemu
安装之后查看会发现有以下应用程序：
ouritsusei@ubuntu:~/Desktop$ qemu-
qemu-img                qemu-nbd                qemu-system-x86_64
qemu-io                 qemu-pr-helper
qemu-make-debian-root   qemu-system-i386

其中，qemu-system-x86_64 用于模拟64位intel架构CPU，qemu-system-i386 模拟32位intel架构CPU，qemu-system-arm 模拟ARM架构(32 位)，qemu-system-aarch64 模拟ARM架构(64位)，等等。
```

5. busybox

```
    BusyBox 是一个集成了三百多个最常用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 Linux 系统的自带的shell。我们在qemu中测试内核时，会将它封装到initramfs使用
```

6. initramfs

Linux系统启动时使用initramfs (initram file system), initramfs可以在启动早期提供一个用户态环境，借助它可以完成一些内核在启动阶段不易完成的工作。当然initramfs是可选的，Linux中的内核编译选项默认开启initrd。在下面的示例情况中你可能要考虑用initramfs。

- 加载模块，比如第三方driver
- 定制化启动过程 (比如打印welcome message等)
- 制作一个非常小的rescue shell
- 任何kernel不能做的，但在用户态可以做的 (比如执行某些命令)

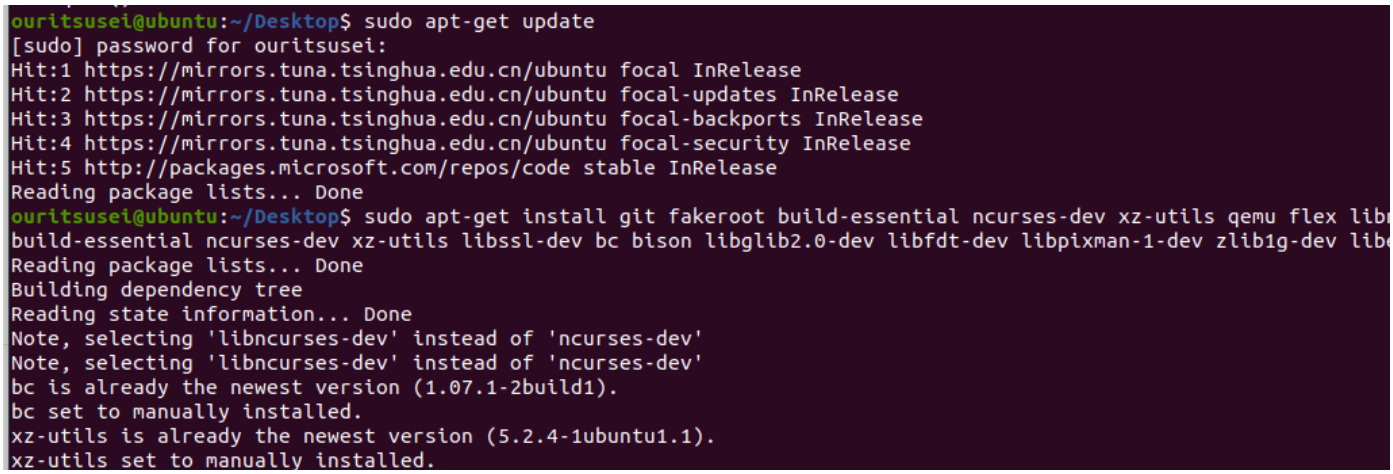
一个initramfs至少要包含一个文件，文件名为/init。内核将这个文件执行起来的进程作为main init进程(pid 1)。当内核挂载initramfs后，文件系统的根分区还没有被mount，这意味着你不能访问文件系统中的任何文件。如果你需要一个shell，必须把shell打包到initramfs中，如果你需要一个简单的工具，比如ls，你也必须把它和它依赖的库或者模块打包到initramfs中。总之，initramfas是一个完全独立运行的体系。

另外initramfs打包的时候，要求打包成压缩的cpio档案。cpio档案可以嵌入到内核image中，也可以作为一个独立的文件在启动的过程中被GRUB load。

三、实验过程

1. 安装依赖库

```
$ sudo apt-get update
$ sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils qemu flex libncurses5-dev fakeroot build-essential ncurses-dev xz-utils libssl-dev bc bison libglib2.0-dev libfdt-dev libpixman-1-dev zlib1g-dev libelf-dev
```



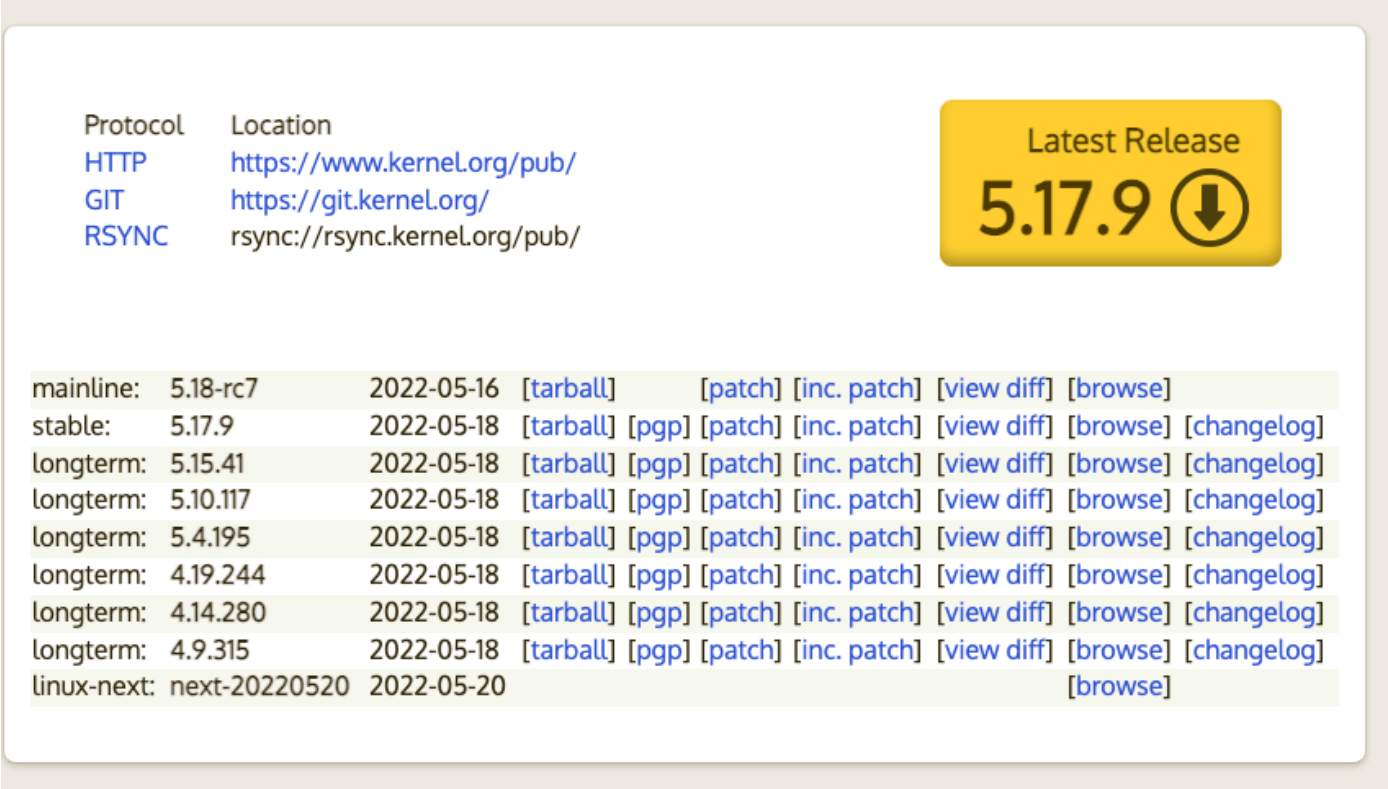
2. 获取内核源码

首先我查看一下我现有Linux系统的内核版本

```
$ uname -srm
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7$ uname -srm
Linux 5.13.0-39-generic x86_64
```

发现是`Linux 5.13.0-39-generic x86_64`，其中`5`为内核版本，`13`为主修订版本，`0-39-generic`为次要修订版本，查看现在最新的为`5.17.9`，我选择直接更新到最新版本。



获取内核源码

```
$ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.17.9.tar.xz
```

```
--2022-05-21 19:40:39--  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.17.9.tar.xz
Connecting to 192.168.178.1:7890... connected.
Proxy request sent, awaiting response... 200 OK
Length: 128449736 (122M) [application/x-xz]
Saving to: ‘linux-5.17.9.tar.xz’

linux-5.17.9.tar.xz                 52%[=============================>                              ]  64.11M  1

2022-05-21 19:41:40 (1.07 MB/s) - Connection closed at byte 67222896. Retrying.

--2022-05-21 19:41:41--  (try: 2)  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.17.9.tar.xz
Connecting to 192.168.178.1:7890... connected.
Proxy request sent, awaiting response... 206 Partial Content
Length: 128449736 (122M), 61226840 (58M) remaining [application/x-xz]
Saving to: ‘linux-5.17.9.tar.xz’

linux-5.17.9.tar.xz                100%[++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++>] 122.50M  1

2022-05-21 19:42:41 (1.00 MB/s) - ‘linux-5.17.9.tar.xz’ saved [128449736/128449736]
```

3. 解压内核源码并配置编译选项

```
$ xz -d linux-5.17.9.tar.xz
$ tar xvf linux-5.17.9.tar
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7$ ls
linux-5.17.9  linux-5.17.9.tar
```

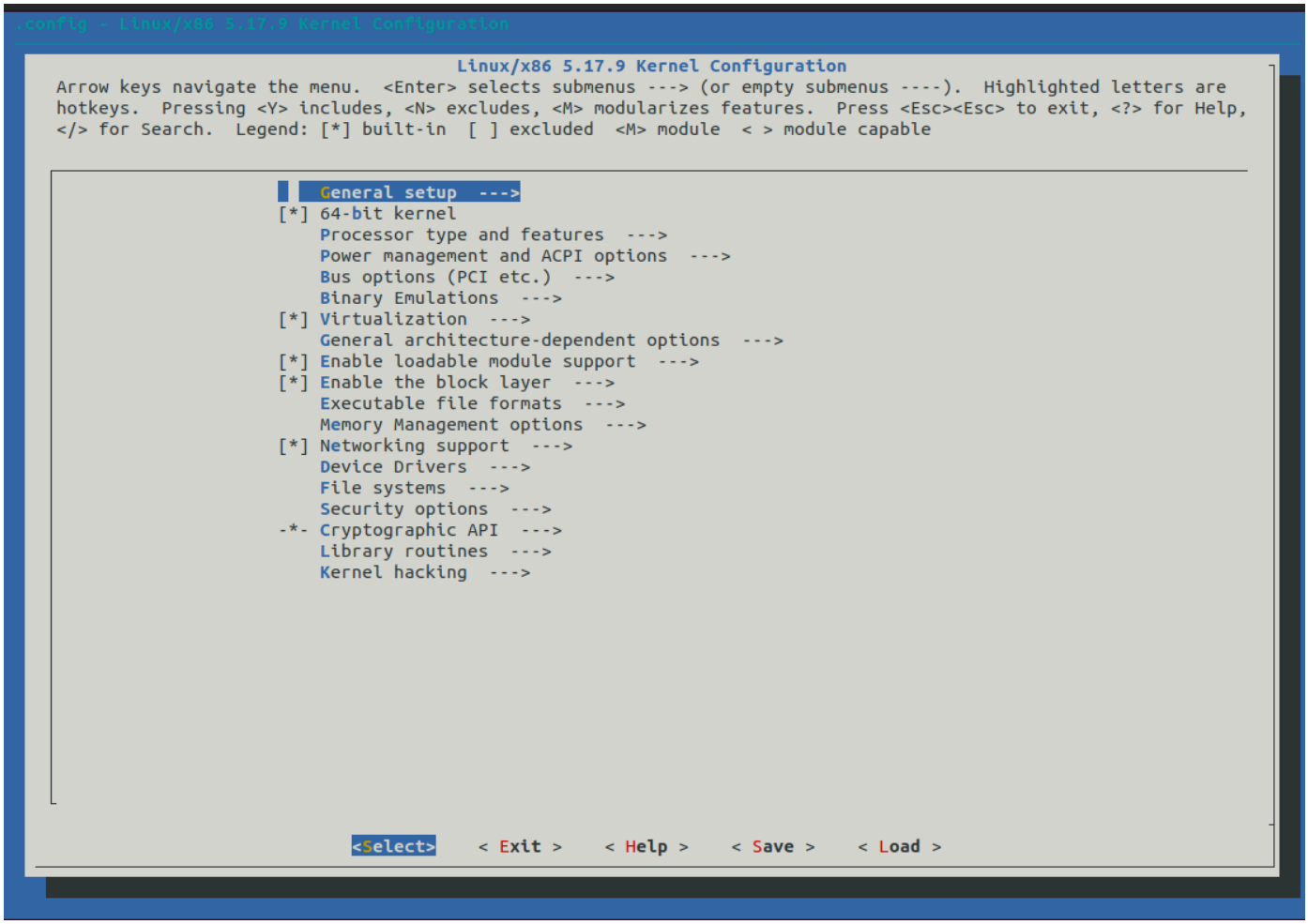
```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ ls
arch    certs     CREDITS  Documentation  fs       init  Kbuild   kernel  LICENSES     Makef
block   COPYING   crypto   drivers        include  ipc   Kconfig  lib     MAINTAINERS  mm
```

在正式编译内核之前，我们首先必须配置需要包含哪些模块。使用cp命令，将当前内核的配置文件拷贝到当前文件夹，然后使用可靠的 `menuconfig` 命令来做任何必要的更改。使用如下命令来完成：

```
$ cp /boot/config-$(uname -r) .config
$ make menuconfig
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ cp /boot/config-$(uname -r) .config
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ make menuconfig
  UPD     scripts/kconfig/mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTLD  scripts/kconfig/mconf
.config:8757:warning: symbol value 'm' invalid for ASHMEM
.config:9809:warning: symbol value 'm' invalid for ANDROID_BINDER_IPC
.config:9810:warning: symbol value 'm' invalid for ANDROID_BINDERFS
.config:10647:warning: symbol value 'm' invalid for CRYPTO_ARCH_HAVE_LIB_BLAKE2S
.config:10648:warning: symbol value 'm' invalid for CRYPTO_LIB_BLAKE2S_GENERIC
configuration written to .config

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```



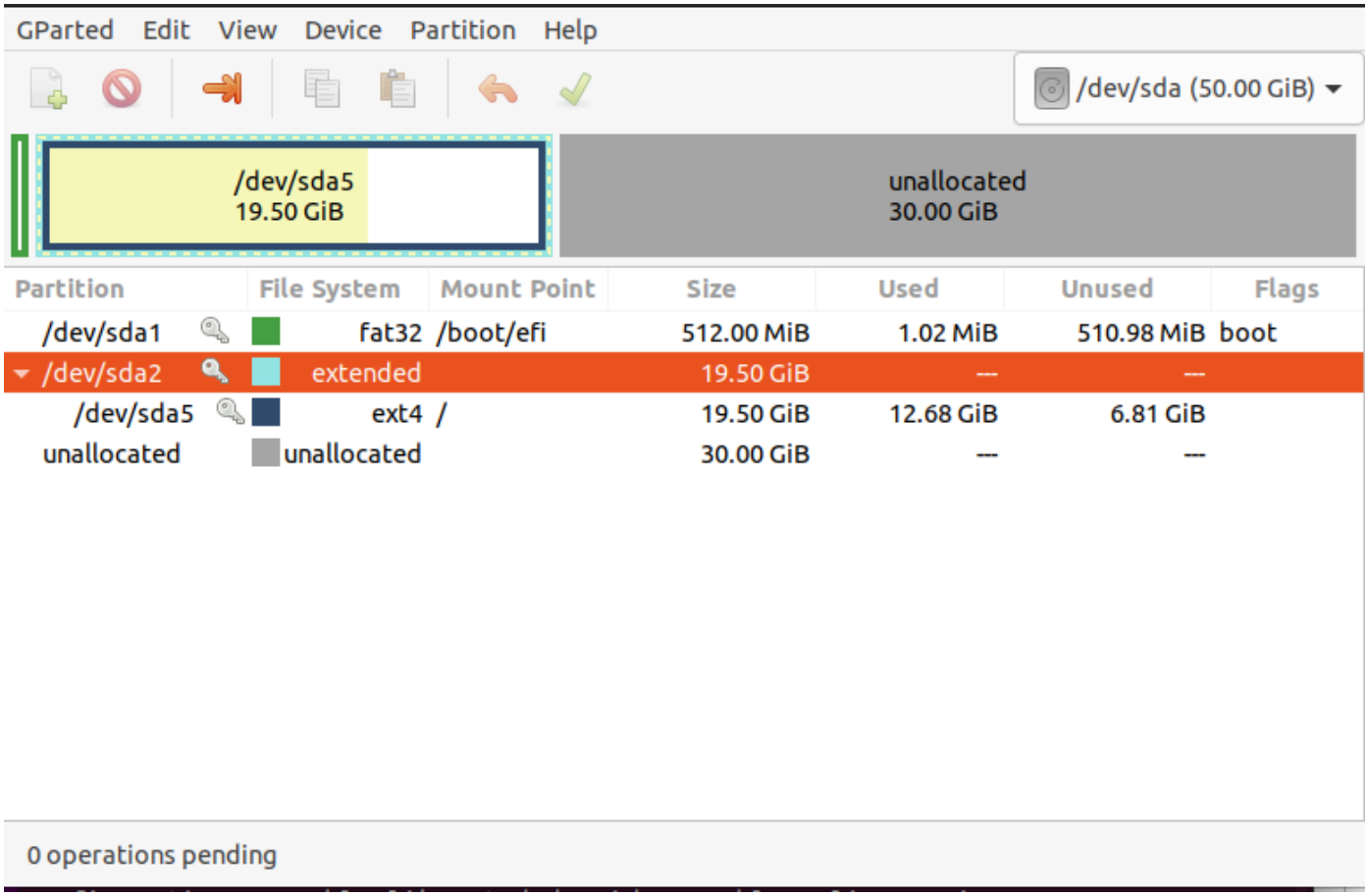
在接下来编译之前一定要确保自己的硬盘空间足够大，查看空间大小

```
$ df -h
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           389M  1.9M  388M   1% /run
/dev/sda5        20G   13G  5.9G  68% /
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/loop0      128K  128K     0 100% /snap/bare/5
/dev/loop1       56M   56M     0 100% /snap/core18/2128
/dev/loop2       62M   62M     0 100% /snap/core20/1434
/dev/loop3       56M   56M     0 100% /snap/core18/2409
/dev/loop4       45M   45M     0 100% /snap/snapd/15534
/dev/loop5      219M  219M     0 100% /snap/gnome-3-34-1804/77
/dev/loop6      249M  249M     0 100% /snap/gnome-3-38-2004/99
/dev/loop7       62M   62M     0 100% /snap/core20/1405
/dev/loop8      219M  219M     0 100% /snap/gnome-3-34-1804/72
/dev/loop9       55M   55M     0 100% /snap/snap-store/558
/dev/loop10      82M   82M     0 100% /snap/gtk-common-themes/1534
/dev/loop11      51M   51M     0 100% /snap/snap-store/547
/dev/loop12      44M   44M     0 100% /snap/snapd/15177
/dev/loop13      66M   66M     0 100% /snap/gtk-common-themes/1519
/dev/sda1       511M  4.0K  511M   1% /boot/efi
tmpfs           389M   36K  389M   1% /run/user/1000
```

发现只有不到10G，肯定是不够用的，所以我又给虚拟机分配了30G，留了35G的空间，肯定够用了

```
$ sudo apt install gparted
$ sudo gparted
```



再次查看`df -h`

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           389M  1.9M  388M   1% /run
/dev/sda5        49G   13G   35G  27% /
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/loop0      128K  128K     0 100% /snap/bare/5
/dev/loop1       56M   56M     0 100% /snap/core18/2128
/dev/loop2       62M   62M     0 100% /snap/core20/1434
/dev/loop3       56M   56M     0 100% /snap/core18/2409
/dev/loop4       45M   45M     0 100% /snap/snapd/15534
/dev/loop5      219M  219M     0 100% /snap/gnome-3-34-1804/77
/dev/loop6      249M  249M     0 100% /snap/gnome-3-38-2004/99
/dev/loop7       62M   62M     0 100% /snap/core20/1405
/dev/loop8      219M  219M     0 100% /snap/gnome-3-34-1804/72
/dev/loop9       55M   55M     0 100% /snap/snap-store/558
/dev/loop10      82M   82M     0 100% /snap/gtk-common-themes/1534
/dev/loop11      51M   51M     0 100% /snap/snap-store/547
/dev/loop12      44M   44M     0 100% /snap/snapd/15177
/dev/loop13      66M   66M     0 100% /snap/gtk-common-themes/1519
/dev/sda1       511M  4.0K  511M   1% /boot/efi
tmpfs           389M   36K  389M   1% /run/user/1000
```

4. 编译

我们首先查看CPU有多少核

```
$ cat /proc/cpuinfo
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
stepping        : 10
cpu MHz         : 2592.000
cache size      : 12288 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
```

`processor = 0`说明虚拟机只有一个核，因为是从0开始计数的，那我们就不用并行编译了，直接用`make`指令编译内核

```
$ make
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ make
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_32_ia32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_64_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_64.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_x32.h
  HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
```

这一步也可以分为两步进行

```
$ make bzimage
$ make modules
```

通过`make`编译完之后，下一步需要编译和安装内核模块

```
$ make modules_install
$ make install
```

编译过程中遇到报错

```
make[1]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'cer
ts/x509_certificate_list'.  Stop.
make: *** [Makefile:1831: certs] Error 2
```

此时我们需要修改`.config`文件

```
$ vim .config
```

修改`CONFIG_SYSTEM_TRUSTED_KEYS`，将其赋空值，找到`CONFIG_SYSTEM_REVOCATION_KEYS`，也将其赋空值。

重新编译

```
$ make
```

编译了5个小时之后，再度报错

```
  GEN     modules.builtin
BTF: .tmp_vmlinux.btf: pahole (pahole) is not available
Failed to generate BTF for vmlinux
Try to disable CONFIG_DEBUG_INFO_BTF
make: *** [Makefile:1155: vmlinux] Error 1
```

查了一下，需要安装`dwarves`

```
$ sudo apt-get install dwarves
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ sudo apt-get install dwarves
[sudo] password for ouritsusei:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfwupdplugin1 linux-headers-5.11.0-27-generic linux-headers-5.13.0-39-generic linux-hwe-5.11-headers-
  linux-image-5.11.0-27-generic linux-image-5.13.0-39-generic linux-modules-5.11.0-27-generic linux-modul
  linux-modules-extra-5.11.0-27-generic linux-modules-extra-5.13.0-39-generic
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  dwarves
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 359 kB of archives.
After this operation, 3,365 kB of additional disk space will be used.
Get:1 http://mirrors.aliyun.com/ubuntu focal-updates/universe amd64 dwarves amd64 1.21-0ubuntu1~20.04 [35
Fetched 359 kB in 1s (685 kB/s)
Selecting previously unselected package dwarves.
(Reading database ... 240681 files and directories currently installed.)
Preparing to unpack .../dwarves_1.21-0ubuntu1~20.04_amd64.deb ...
Unpacking dwarves (1.21-0ubuntu1~20.04) ...
Setting up dwarves (1.21-0ubuntu1~20.04) ...
Processing triggers for man-db (2.9.1-1) ...
```

重新编译

```
$ make
```

再次报错

```
  BTFIDS  vmlinux
FAILED: load BTF from vmlinux: No such file or directory
make: *** [Makefile:1155: vmlinux] Error 255
make: *** Deleting file 'vmlinux'
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$
```

修改`CONFIG_DEBUG_INFO=y`为`CONFIG_DEBUG_INFO=n`，重新编译

```
$ make
```

给出选择提示信息

```
  SYNC    include/config/auto.conf.cmd
*
* Restart config...
*
*
* Compile-time checks and compiler options
*
Compile the kernel with debug info (DEBUG_INFO) [Y/n/?] y
  Reduce debugging information (DEBUG_INFO_REDUCED) [N/y/?] n
  Compressed debugging information (DEBUG_INFO_COMPRESSED) [N/y/?] n
  Produce split debuginfo in .dwo files (DEBUG_INFO_SPLIT) [N/y/?] n
  DWARF version
    1. Rely on the toolchain's implicit default DWARF version (DEBUG_INFO_DWARF_TOOLCHAIN_DEFAULT)
  > 2. Generate DWARF Version 4 debuginfo (DEBUG_INFO_DWARF4)
    3. Generate DWARF Version 5 debuginfo (DEBUG_INFO_DWARF5) (NEW)
  choice[1-3?]: 1
```

再次报错

```
  ZSTD22  arch/x86/boot/compressed/vmlinux.bin.zst
/bin/sh: 1: zstd: not found
make[2]: *** [arch/x86/boot/compressed/Makefile:139: arch/x86/boot/compressed/vmlinux.bin.zst] Error 127
make[2]: *** Deleting file 'arch/x86/boot/compressed/vmlinux.bin.zst'
make[1]: *** [arch/x86/boot/Makefile:115: arch/x86/boot/compressed/vmlinux] Error 2
make: *** [arch/x86/Makefile:269: bzImage] Error 2
```

安装`zstd`后重新编译

```
$ sudo apt install zstd
$ make
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ sudo apt install zstd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfwupdplugin1
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  zstd
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 343 kB of archives.
After this operation, 1,592 kB of additional disk space will be used.
Get:1 http://mirrors.aliyun.com/ubuntu focal-security/universe amd64 zstd amd64 1.4.4+dfsg-3ubuntu0.1 [343 kB]
Fetched 343 kB in 1s (260 kB/s)
Selecting previously unselected package zstd.
(Reading database ... 167533 files and directories currently installed.)
Preparing to unpack .../zstd_1.4.4+dfsg-3ubuntu0.1_amd64.deb ...
Unpacking zstd (1.4.4+dfsg-3ubuntu0.1) ...
Setting up zstd (1.4.4+dfsg-3ubuntu0.1) ...
Processing triggers for man-db (2.9.1-1) ...
```

终于编译完成了

```
  CC [M]  sound/x86/snd-hdmi-lpe-audio.mod.o
  LD [M]  sound/x86/snd-hdmi-lpe-audio.ko
  CC [M]  sound/xen/snd_xen_front.mod.o
  LD [M]  sound/xen/snd_xen_front.ko
  GEN     scripts/gdb/linux/constants.py
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$
```

5. 使用`qemu`测试

> Busybox 是一个集成了三百多个最常用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。

接下来使用`qemu`测试一下编译出来的内核是不是有问题，下载`busybox`，我选择的是最新版的`1.35.0`

```
$ wget https://busybox.net/downloads/busybox-1.35.0.tar.bz2
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7$ wget https://busybox.net/downloads/busybox-1.35.0.tar.bz2
--2022-05-22 23:09:26--  https://busybox.net/downloads/busybox-1.35.0.tar.bz2
Resolving busybox.net (busybox.net)... 140.211.167.122
Connecting to busybox.net (busybox.net)|140.211.167.122|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2480624 (2.4M) [application/x-bzip2]
Saving to: ‘busybox-1.35.0.tar.bz2’

busybox-1.35.0.tar.bz2                100%[===================================================================

2022-05-22 23:09:32 (475 KB/s) - ‘busybox-1.35.0.tar.bz2’ saved [2480624/2480624]

ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7$ ls
busybox-1.35.0.tar.bz2  linux-5.17.9  linux-5.17.9.tar
```

解压并修改配置文件将`Settings`-->`Build static binary(no shared libs)`选中

```
$ tar xvjf busybox-1.35.0.tar.bz2
```

```
$ cd busybox-1.35.0
$ make menuconfig
```

进行编译

```
$ make
$ make install
```

```
./_install//usr/sbin/ubidetach -> ../../bin/busybox
./_install//usr/sbin/ubimkvol -> ../../bin/busybox
./_install//usr/sbin/ubirename -> ../../bin/busybox
./_install//usr/sbin/ubirmvol -> ../../bin/busybox
./_install//usr/sbin/ubirsvol -> ../../bin/busybox
./_install//usr/sbin/ubiupdatevol -> ../../bin/busybox
./_install//usr/sbin/udhcpd -> ../../bin/busybox


---------------------------------------------------
You will probably need to make your busybox binary
setuid root to ensure all configured applets will
work properly.
---------------------------------------------------

ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0$
```

编译完成之后会生成`_install`目录，其内容如下：

```
$ ls
$ ls _install/
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0$ ls
applets             busybox_unstripped.map  e2fsprogs          _install     Makefile                NOFORK_NOEXEC.lst       selinux
applets_sh          busybox_unstripped.out  editors            INSTALL      Makefile.custom         NOFORK_NOEXEC.sh        shell
arch                Config.in               examples           klibc-utils  Makefile.flags          printutils              size_single_applets.sh
archival            configs                 findutils          libbb        Makefile.help           procps                  sysklogd
AUTHORS             console-tools           include            libpwdgrp    make_single_applets.sh  qemu_multiarch_testing  testsuite
busybox             coreutils               init               LICENSE      miscutils               README                  TODO
busybox.links       debianutils             initramfs          loginutils   modutils                runit                   TODO_unicode
busybox_unstripped  docs                    initramfs.cpio.gz  mailutils    networking              scripts                 util-linux
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0$ ls _install/
bin  linuxrc  sbin  usr
```

> Linux启动阶段，boot loader加载完内核文件vmlinuz之后，便开始挂载磁盘根文件系统。挂载操作需要磁盘驱动，所以挂载前要先加载驱动。但是驱动位于`/lib/modules`，不挂载磁盘就访问不到，形成了一个死循环。`initramfs`根文件系统就可以解决这个问题，其中包含必要的设备驱动和工具，boot loader会加载initramfs到内存中，内核将其挂载到根目录，然后运行`/init`初始化脚本，去挂载真正的磁盘根文件系统。

接下来我们创建`inittramfs`

```
mkdir initramfs
cd initramfs
cp ../_install/* -rf ./
mkdir dev proc sys
sudo cp -a /dev/{null,console,tty,tty1,tty2,tty3,tty4} dev/
rm linuxrc
vim init
chmod a+x init
```

```
or available locally via: info '(coreutils) cp invocation'
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ cp ../_install/* -rf ./
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ ls
bin  linuxrc  sbin  usr
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ sudo cp -a /dev/{null, console, tty, tty1, tty2, tty3, tty4} dev/
[sudo] password for ouritsusei:
cp: target 'dev/' is not a directory
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ mkdir dev proc sys
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ sudo cp -a /dev/{null, console, tty, tty1, tty2, tty3, tty4} dev/
cp: cannot stat '/dev/{null,': No such file or directory
cp: cannot stat 'console,': No such file or directory
cp: cannot stat 'tty,': No such file or directory
cp: cannot stat 'tty1,': No such file or directory
cp: cannot stat 'tty2,': No such file or directory
cp: cannot stat 'tty3,': No such file or directory
cp: cannot stat 'tty4}': No such file or directory
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ sudo cp -a /dev/{null,console,tty,tty1,tty2,tty3,tty4} dev/
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ ls
bin  dev  linuxrc  proc  sbin  sys  usr
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ cd dev
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs/dev$ ls
console  null  tty  tty1  tty2  tty3  tty4
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs/dev$ cd ..
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ ls
bin  dev  linuxrc  proc  sbin  sys  usr
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ rm linuxrc
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ vim init
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ ls
bin  dev  init  proc  sbin  sys  usr
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$ chmod a+x init
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/busybox-1.35.0/initramfs$
```

在`init`文件中写的内容如下：

```
#!/bin/busybox sh
mount -t proc none /proc
mount -t sysfs none /sys

exec /sbin/init
```

在创建的initramfs中包含busybox可执行程序、必须的设备文件、启动脚本`init`，且`init`只挂载了虚拟文件系统`procfs`和`sysfs`，没有挂载磁盘根文件系统，所有操作都在内存中进行。

最后打包`initramfs`

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

接下来启动内核：

```
$ qemu-system-i386 -s -kernel ../linux-5.17.9/arch/x86_64/boot/bzImage -initrd
initramfs.cpio.gz -nographic -append "console=ttyS0"
```

```
-m：虚拟机内存大小
-kernel：内存镜像路径
-initrd：磁盘镜像路径
-append
    nokalsr：关闭内核地址随机化，方便我们进行调试
    rdinit：指定初始启动进程，`/sbin/init`进程会默认以`/etc/init.d/rcS`作为启动脚本
    loglevel=3 & quiet：不输出log
    console=ttyS0：指定终端为`/dev/ttyS0`，这样一启动就能进入终端界面
-monitor：将监视器重定向到主机设备`/dev/null`
-cpu：设置CPU安全选项，在这里开启了smep保护
-s：相当于-gdb tcp::1234的简写（也可以直接这么写），后续我们可以通过gdb连接本地端口进行调试
```

```
SeaBIOS (version 1.13.0-1ubuntu1.1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+07F8C8B0+07ECC8B0 CA00



Booting from ROM...
This kernel requires an x86-64 CPU, but only detected an i686 CPU.
Unable to boot - please use a kernel appropriate for your CPU.
```

报错，发现是自己模拟的CPU搞错了，接下来启动内核。

```
$ qemu-system-x86_64 -s -kernel ../linux-5.17.9/arch/x86_64/boot/bzImage -initrd
initramfs.cpio.gz -nographic -append "console=ttyS0"
```

```
[    2.199293] evm: HMAC attrs: 0x1
[    2.210862] sr 1:0:0:0: [sr0] scsi3-mmc drive: 4x/4x cd/rw xa/form2 tray
[    2.211230] cdrom: Uniform CD-ROM driver Revision: 3.20
[    2.215449] PM:   Magic number: 6:798:974
[    2.221130] RAS: Correctable Errors collector initialized.
[    2.229509] sr 1:0:0:0: Attached scsi generic sg0 type 5
[    2.246176] Freeing unused decrypted memory: 2036K
[    2.260893] Freeing unused kernel image (initmem) memory: 2732K
[    2.261318] Write protecting the kernel read-only data: 24576k
[    2.265274] Freeing unused kernel image (text/rodata gap) memory: 2036K
[    2.266269] Freeing unused kernel image (rodata/data gap) memory: 536K
[    2.410613] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[    2.411496] Run /init as init process
can't run '/etc/init.d/rcS': No such file or directory

Please press Enter to activate this console. [    2.530394] tsc: Refined TSC clocksource calibration: 2592.856 MHz
[    2.531622] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x255fdfcad54, max_idle_ns: 440795331766 ns
[    2.534439] clocksource: Switched to clocksource tsc

/ # ls
bin   dev   init  proc  root  sbin  sys   usr
/ #
```

成功启动，说明没问题，接下来替换内核。

6. 替换内核

安装模块

```
$ sudo make modules_install
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ sudo make modules_install
[sudo] password for ouritsusei:
arch/x86/Makefile:154: CONFIG_X86_X32 enabled but no binutils support
  INSTALL /lib/modules/5.17.9/kernel/arch/x86/crypto/aegis128-aesni.ko
  SIGN    /lib/modules/5.17.9/kernel/arch/x86/crypto/aegis128-aesni.ko
  INSTALL /lib/modules/5.17.9/kernel/arch/x86/crypto/aesni-intel.ko
  SIGN    /lib/modules/5.17.9/kernel/arch/x86/crypto/aesni-intel.ko
  INSTALL /lib/modules/5.17.9/kernel/arch/x86/crypto/blake2s-x86_64.ko
```

安装内核

```
$ sudo make install
```

```
  DEPMOD  /lib/modules/5.17.9
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ sudo make install
arch/x86/Makefile:154: CONFIG_X86_X32 enabled but no binutils support
sh ./arch/x86/boot/install.sh 5.17.9 \
        arch/x86/boot/bzImage System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.17.9 /boot/vmlinuz-5.17.9
update-initramfs: Generating /boot/initrd.img-5.17.9
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.17.9 /boot/vmlinuz-5.17.9
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.17.9 /boot/vmlinuz-5.17.9
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.17.9 /boot/vmlinuz-5.17.9
I: /boot/initrd.img.old is now a symlink to initrd.img-5.13.0-44-generic
I: /boot/initrd.img is now a symlink to initrd.img-5.17.9
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.17.9 /boot/vmlinuz-5.17.9
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.17.9
Found initrd image: /boot/initrd.img-5.17.9
Found linux image: /boot/vmlinuz-5.13.0-44-generic
Found initrd image: /boot/initrd.img-5.13.0-44-generic
Found linux image: /boot/vmlinuz-5.13.0-41-generic
Found initrd image: /boot/initrd.img-5.13.0-41-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

启用内核作为引导

```
$ sudo update-initramfs -c -k 5.17.9
```

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$  sudo update-initramfs -c -k 5.17.9
update-initramfs: Generating /boot/initrd.img-5.17.9
```

更新`grub`

```
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.17.9
Found initrd image: /boot/initrd.img-5.17.9
Found linux image: /boot/vmlinuz-5.13.0-44-generic
Found initrd image: /boot/initrd.img-5.13.0-44-generic
Found linux image: /boot/vmlinuz-5.13.0-41-generic
Found initrd image: /boot/initrd.img-5.13.0-41-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

接下来就可以重启看看内核版本了

然后发现开不了机了

因为中间编译内核模块的时候有提示空间不足，所以首先进行磁盘拓展重新编译内核模块，发现仍然不行。

之后查阅了大量的资料

发现在编译内核过程中，安装内核模块时未使用`INSTALL_MOD_STRIP=1`会导致`initrd`文件过大，而`Ubuntu 20.04`所用的`Grub 2.04`无法支持过大的`initrd`文件（如500M），会导致内核启动一直卡在`Loading initial randisk`.

参考资料：https://superuser.com/questions/705121/why-is-install-mod-strip-not-on-by-default

解决办法：

重新安装模块，并且加上`INSTALL_MOD_STRIP=1`

```
$ sudo make INSTALL_MOD_STRIP=1 modules_install
```

经过不懈努力，接下来看一下我们的成果吧

```
$ uname -arm
```

```
ouritsusei@ubuntu:~/Desktop$ uname -arm
Linux ubuntu 5.17.9 #3 SMP PREEMPT Sun May 22 09:42:07 PDT 2022 x86_64 x86_64 x
86_64 GNU/Linux
```

成功替换内核（虽然说遇到了各种艰难险阻

四、实验结果与分析

查询最后启动的内核版本

```
$ uname -arm
```

```
ouritsusei@ubuntu:~/Desktop$ uname -arm
Linux ubuntu 5.17.9 #3 SMP PREEMPT Sun May 22 09:42:07 PDT 2022 x86_64 x86_64 x
86_64 GNU/Linux
```

可以看到我们的内核版本已经被替换为`5.17.9`

五、问题总结

1. 空间不足

编译内核模块时提示空间不足，关闭虚拟机进行硬盘拓展后重新编译

2. 编译内核时遇到报错

```
make[1]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'cer
ts/x509_certificate_list'.  Stop.
make: *** [Makefile:1831: certs] Error 2
```

这是因为直接使用的现有的配置文件，没有进行修改。修改`CONFIG_SYSTEM_TRUSTED_KEYS`，将其赋空值，找到`CONFIG_SYSTEM_REVOCATION_KEYS`，也将其赋空值。

3. 编译内核时遇到报错

```
  BTFIDS  vmlinux
FAILED: load BTF from vmlinux: No such file or directory
make: *** [Makefile:1155: vmlinux] Error 255
make: *** Deleting file 'vmlinux'
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$
```

这是没有`dwarves`，`sudo apt install dwarves`进行下载

4. 编译内核时遇到报错

```
  BTFIDS  vmlinux
FAILED: load BTF from vmlinux: No such file or directory
make: *** [Makefile:1155: vmlinux] Error 255
make: *** Deleting file 'vmlinux'
ouritsusei@ubuntu:~/Desktop/OS_expeirment/ex_7/linux-5.17.9$
```

还是因为直接使用现有的配置文件，没有进行相应的修改。修改`CONFIG_DEBUG_INFO=y`为`CONFIG_DEBUG_INFO=n`

5. 编译内核时遇到报错

```
  ZSTD22  arch/x86/boot/compressed/vmlinux.bin.zst
/bin/sh: 1: zstd: not found
make[2]: *** [arch/x86/boot/compressed/Makefile:139: arch/x86/boot/compressed/vmlinux.bin.zst] Error 127
make[2]: *** Deleting file 'arch/x86/boot/compressed/vmlinux.bin.zst'
make[1]: *** [arch/x86/boot/Makefile:115: arch/x86/boot/compressed/vmlinux] Error 2
make: *** [arch/x86/Makefile:269: bzImage] Error 2
```

这是因为没有`zstd`，`sudo apt install zstd`进行下载

6. `initrd`过大

在编译内核过程中，安装内核模块时未使用`INSTALL_MOD_STRIP=1`会导致`initrd`文件过大，而`Ubuntu 20.04`所用的`Grub 2.04`无法支持过大的`initrd`文件（如500M），会导致内核启动一直卡在`Loading initial randisk`，`$ sudo make INSTALL_MOD_STRIP=1 modules_install`即可解决。

六、参考文献

[1] allocation errors with secureboot grub version https://bugzilla.redhat.com/show_bug.cgi?id=1572126

[2] Grub 2.04启动内核卡在“loading initial ramdisk” https://codeantenna.com/a/ZkNULLTebn

[3] Why is INSTALL_MOD_STRIP not on by default? https://superuser.com/questions/705121/why-is-install-mod-strip-not-on-by-default

[4] Linux内核编译 https://github.com/arttnba3/XDU-SCE_OS-Experiment_2021/tree/main/Exp-7

[5] 一文读懂QEMU虚拟机 https://fonttian.blog.csdn.net/article/details/103924589

[6] QEMU入门指南 https://blog.csdn.net/FontThrone/article/details/104157859

[7] Linux内核编译很简单，6步编译一个自己的内核 https://os.51cto.com/article/663841.html

[8] 如何编译 Linux 内核 https://zhuanlan.zhihu.com/p/37164435

[9] Linux内核编译与安装 https://www.linuxprobe.com/linux-kernel-compilation.html

[10] Linux 内核编译步骤及配置详解 https://www.cnblogs.com/xiaocen/p/3717993.html

[11] 如何编译 Linux 内核 https://linux.cn/article-9665-1.html

[12] Kernel Panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0) https://askubuntu.com/questions/41930/kernel-panic-not-syncing-vfs-unable-to-mount-root-fs-on-unknown-block0-0

[13] Boot hangs at "loading initial ramdisk" https://bugs.launchpad.net/ubuntu/+source/intel-microcode/+bug/1853369

[14] qemu运行linux内核 https://www.cnblogs.com/tbolp/p/15219547.html

[15] Linux内存管理之一 —— 环境搭建（在QEMU上运行Linux 5.4.0）https://luomuxiaoxiao.com/?p=743

[16] QEMU运行Linux Kernel环境配置 https://www.cxyzjd.com/article/HaoBBNuanMM/106625017

[17] QEMU + Busybox 模拟 Linux 内核环境 https://www.v4ler1an.com/2020/12/qemu/

[18] 基于Qemu搭建x86_64虚拟环境运行Linux内核 https://blog.csdn.net/fantasy_wxe/article/details/108418822