

北京理工大学

操作系统课程设计

实验一、编译 Linux 内核	Experiment 1, Compiling the Linux Kernel
--------------------	------------------------------------------------

学院： 计算机学院

专业： 计算机科学与技术

学生姓名： 夏奇拉

学号： 1820171025

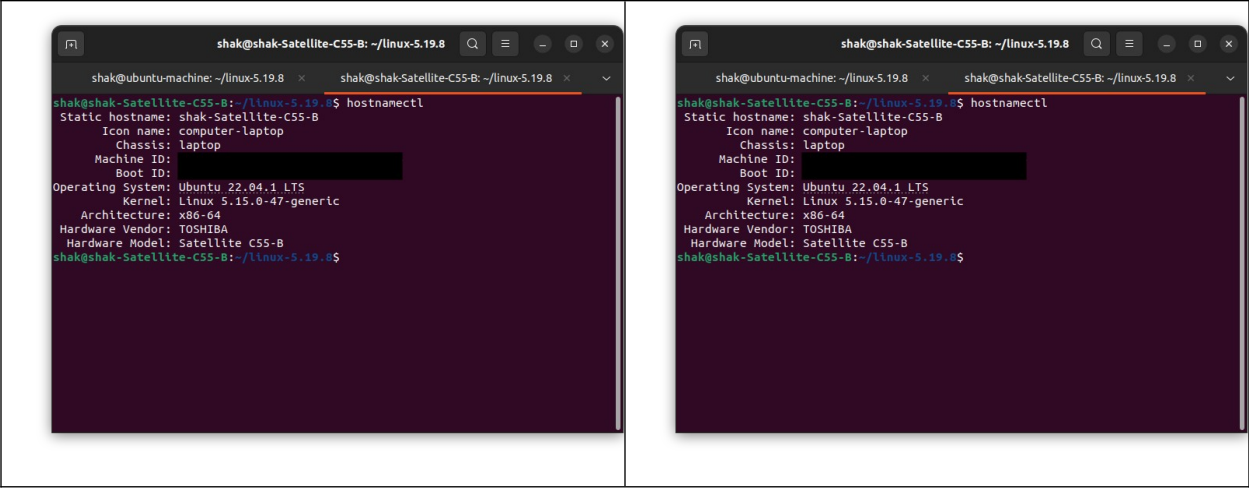
班级： 07111705

实验目的	Experiment Purpose
<p>在内核中加入本人学号，具体做法：</p> <p>修改 makefile 文件中有字段 EXTRAVERSION，通过修改它就可以使得自己编译的 linux 内核版本信息改变。使用文本编辑器即可进行修改。</p> <p>样例：</p> <p>EXTRAVERSION = -1120131860</p> <p>注意：不要忘记学号前面的短横线(半角)</p>	<p>The experiment asks that one</p> <ol style="list-style-type: none"> 1. Compiles their own Linux kernel 2. adds their student ID to the kernel by modifying the EXTRAVERSION field in the makefile. <p>Example:</p> <p>EXTRAVERSION = -1120131860</p> <p>Note: Don't forget the dash in front of the student number</p> <p>*Modifications can be made using a text editor.</p>

实验内容	Experiment Content
<ol style="list-style-type: none"> 1. 从源码编译安装 Linux 内核 2. 修改内核，使其运行时版本字符串中包含字符“=1820171025” <p>术语定义</p> <p>Linux：是一种操作系统，一种控制系统硬件组件的软件。操作系统由引导加载程序、内核、网络、外壳等组成。</p> <p>内核是系统的核心，管理着 CPU、内存和外围设备。它是用 C 和汇编编程语言编写的。</p> <p>Source 是指内核源代码，可以在 kernel.org 上找到，kernel.org 是由 Linux Kernel Organization 运营的网站。</p> <p>自己编译 Linux Kernel 的原因：</p> <ul style="list-style-type: none"> • 了解它是如何完成的 • 启用标准内核中没有的硬件支持 • 启用在标准内核中找不到的选项 • 每次实现自己的系统调用或修改内核源代码 	<ol style="list-style-type: none"> 1. Compile and install Linux kernel from source 2. Modify the kernel so that running it has the characters “=1820171025” in the version string <p>Definition of terms</p> <p>Linux: is an operating system, a piece of software that controls the hardware components of a system. An operating system consists of a bootloader, kernel, networking, shell, etc.</p> <p>The Kernel is the core of the system and manages the CPU, memory and peripheral devices. It is written in C and Assembly programming languages.</p> <p>Source refers to the kernel source code which can be found at kernel.org, a website run by the Linux Kernel Organization.</p> <p>Reasons for compiling the Linux Kernel yourself:</p>

<p>码时； 您将需要重新编译内核以实现更改。</p> <p>从源代码编译和安装 Linux 内核涉及</p> <ul style="list-style-type: none"> • 从 kernel.org 下载最新的内核 • 使用 apt-get 安装所需的软件包 • 配置内核和模块 • 编译和构建内核 • 更新 grub 配置 • 重新启动系统 	<ul style="list-style-type: none"> • to learn how it's done • to enable hardware support not found in the standard kernel • to enable options not found in the standard kernel • each time you implement your own system call or modify kernel source code; you will need to recompile the kernel to implement the changes. <p>Compiling and installing the Linux kernel from source involves</p> <ul style="list-style-type: none"> • Downloading the latest kernel from kernel.org • Installing required packages using apt-get • Configuring the kernel and modules • Compiling and building the kernel • Updating the grub configuration • Rebooting the system
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

实验环境	Experiment Environment
<p>先决条件：</p> <p>基于 Linux 的操作系统</p> <p>12 GB 可用磁盘空间</p> <p>用于下载源代码的 Internet 连接</p> <p>还有很多时间，至少 90 分钟</p>	<p>Prerequisites:</p> <ul style="list-style-type: none"> • A Linux based operating system • 12 GB of free disk space • An internet connection to download the source code • And a lot of time, at least 90 minutes



操作方法和实验步骤
(程序设计与实现)

Operating Methods and Experimental Steps
(Program design and Implementation)

第一步：下载源代码

Step 1: Download the source code

Linux 内核源代码可以在 kernel.org 上找到。这里我们可以通过点击“Latest Release”来获取内核的 URL。

The Linux kernel source code can be found at kernel.org. Here we can get the URL of the kernel by clicking “Latest Release”.



网址：<https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz>

URL:
<https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz>

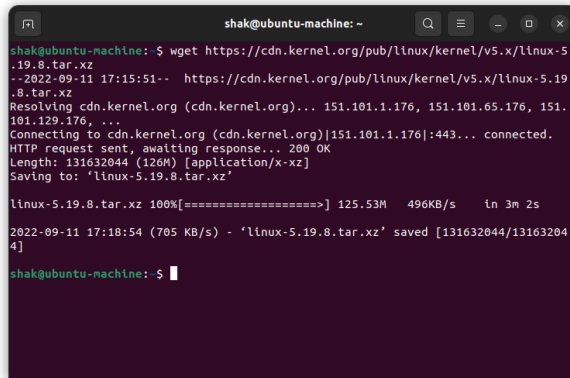
然后我们打开终端，使用 wget 命令下载 Linux 内核源代码：

We then open the terminal and use the wget command to download the Linux kernel source

wget

`https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz`

该命令的输出显示“下载完成时已保存的消息。”



```
shak@ubuntu-machine: ~  
shak@ubuntu-machine:~$ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz  
--2022-09-11 17:15:51-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz  
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.1.176, 151.101.65.176, 151.101.129.176, ...  
Connecting to cdn.kernel.org (cdn.kernel.org)[151.101.1.176]:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 131632044 (126M) [application/x-xz]  
Saving to: 'linux-5.19.8.tar.xz'  
  
linux-5.19.8.tar.xz 100%[=====] 125.53M 496KB/s in 3m 2s  
2022-09-11 17:18:54 (705 KB/s) - 'linux-5.19.8.tar.xz' saved [131632044/131632044]  
shak@ubuntu-machine:~$
```

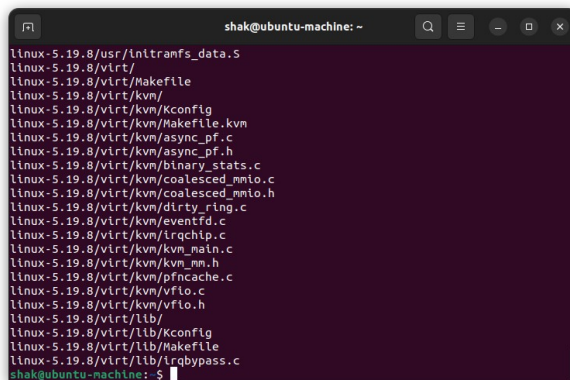
注意：Wget 是一个免费的 GNU 命令行实用工具，用于从 Internet 下载文件。

第二步：提取源代码

文件准备好后，运行 `tar` 命令提取源代码。

`tar xvf linux-5.19.8.tar.xz`

输出显示提取的内核源代码。



```
shak@ubuntu-machine: ~  
linux-5.19.8/usr/initramfs_data.S  
linux-5.19.8/virt/  
linux-5.19.8/virt/Makefile  
linux-5.19.8/virt/kvm/  
linux-5.19.8/virt/kvm/kconfig  
linux-5.19.8/virt/kvm/Makefile.kvm  
linux-5.19.8/virt/kvm/async_pf.c  
linux-5.19.8/virt/kvm/async_pf.h  
linux-5.19.8/virt/kvm/binary_stats.c  
linux-5.19.8/virt/kvm/coalesced_mmio.c  
linux-5.19.8/virt/kvm/coalesced_mmio.h  
linux-5.19.8/virt/kvm/dirty_ring.c  
linux-5.19.8/virt/kvm/eventfd.c  
linux-5.19.8/virt/kvm/irqchip.c  
linux-5.19.8/virt/kvm/kvm_main.c  
linux-5.19.8/virt/kvm/kvm_mm.h  
linux-5.19.8/virt/kvm/pfncache.c  
linux-5.19.8/virt/kvm/vfio.c  
linux-5.19.8/virt/kvm/vfio.h  
linux-5.19.8/virt/llb/  
linux-5.19.8/virt/llb/kconfig  
linux-5.19.8/virt/llb/Makefile  
linux-5.19.8/virt/llb/irqbypass.c  
shak@ubuntu-machine:~$
```

注意：Tar 命令是磁带存档的缩写，用于创建、提取和维护存档（压缩）文件。语法采用三个参数：

`$ tar [选项] [归档文件] [要归档的目录/文件]`

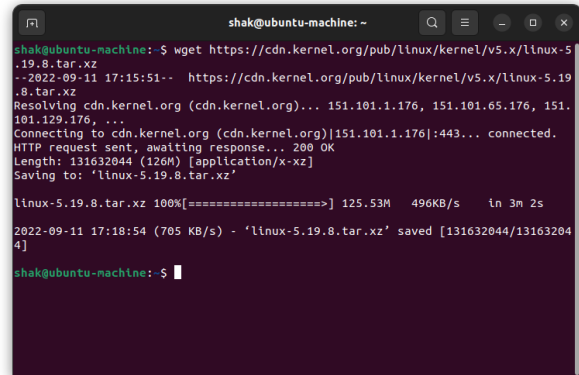
Xvf 是 Unix 风格的简短方法，用于实现 -

code:

`wget`

`https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz`

The output of the command shows the “saved message when the download is complete.



```
shak@ubuntu-machine: ~  
shak@ubuntu-machine:~$ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz  
--2022-09-11 17:15:51-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.8.tar.xz  
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.1.176, 151.101.65.176, 151.101.129.176, ...  
Connecting to cdn.kernel.org (cdn.kernel.org)[151.101.1.176]:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 131632044 (126M) [application/x-xz]  
Saving to: 'linux-5.19.8.tar.xz'  
  
linux-5.19.8.tar.xz 100%[=====] 125.53M 496KB/s in 3m 2s  
2022-09-11 17:18:54 (705 KB/s) - 'linux-5.19.8.tar.xz' saved [131632044/131632044]  
shak@ubuntu-machine:~$
```

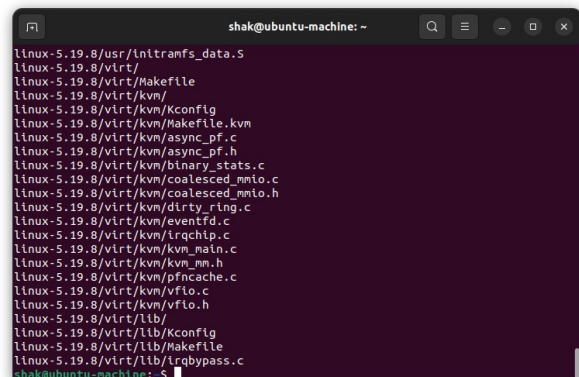
Note: Wget is a free GNU command-line utility tool used to download files from the internet.

Step 2: Extract the source code

When the file is ready, run the `tar` command to extract the source code.

`tar xvf linux-5.19.8.tar.xz`

The output displays the extracted kernel source code.



```
shak@ubuntu-machine: ~  
linux-5.19.8/usr/initramfs_data.S  
linux-5.19.8/virt/  
linux-5.19.8/virt/Makefile  
linux-5.19.8/virt/kvm/  
linux-5.19.8/virt/kvm/kconfig  
linux-5.19.8/virt/kvm/Makefile.kvm  
linux-5.19.8/virt/kvm/async_pf.c  
linux-5.19.8/virt/kvm/async_pf.h  
linux-5.19.8/virt/kvm/binary_stats.c  
linux-5.19.8/virt/kvm/coalesced_mmio.c  
linux-5.19.8/virt/kvm/coalesced_mmio.h  
linux-5.19.8/virt/kvm/dirty_ring.c  
linux-5.19.8/virt/kvm/eventfd.c  
linux-5.19.8/virt/kvm/irqchip.c  
linux-5.19.8/virt/kvm/kvm_main.c  
linux-5.19.8/virt/kvm/kvm_mm.h  
linux-5.19.8/virt/kvm/pfncache.c  
linux-5.19.8/virt/kvm/vfio.c  
linux-5.19.8/virt/kvm/vfio.h  
linux-5.19.8/virt/llb/  
linux-5.19.8/virt/llb/kconfig  
linux-5.19.8/virt/llb/Makefile  
linux-5.19.8/virt/llb/irqbypass.c  
shak@ubuntu-machine:~$
```

Note: Tar commands, short for tape archive, are used to create, extract and maintain archived (compressed) files. The syntax takes

`extract -verbose -file。`

X = 提取档案,

V = 显示详细信息

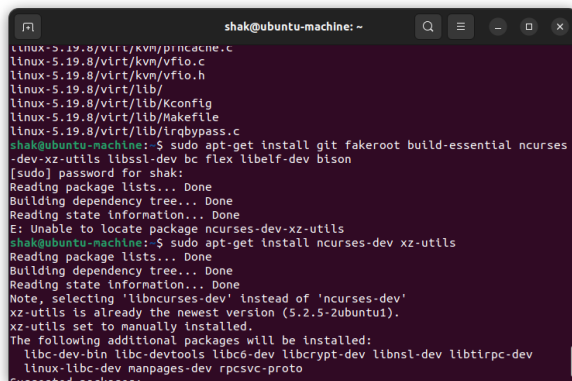
F = 指定文件名

详细信息提供有关正在执行的任务的其他详细信息。

第 3 步: 安装所需的软件包

为了成功编译内核, 需要特定的软件包。要安装这些软件包, 我们运行以下命令:

```
sudo apt-get install git fakeroot build-essential
ncurses-dev-xz-utils libssl-dev bc flex libelf-dev
bison
```



```
shak@ubuntu-machine: ~
linux-5.19.8/virt/kvm/pnucacne.c
linux-5.19.8/virt/kvm/vfio.c
linux-5.19.8/virt/kvm/vfio.h
linux-5.19.8/virt/lib/
linux-5.19.8/virt/lib/Kconfig
linux-5.19.8/virt/lib/Makefile
linux-5.19.8/virt/lib/irqbypass.c
shak@ubuntu-machine: ~$ sudo apt-get install git fakeroot build-essential ncurses
-dev-xz-utils libssl-dev bc flex libelf-dev bison
[sudo] password for shak:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package ncurses-dev-xz-utils
shak@ubuntu-machine: ~$ sudo apt-get install ncurses-dev xz-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'libncurses-dev' instead of 'ncurses-dev'
xz-utils is already the newest version (5.2.5-2ubuntu1).
xz-utils set to manually installed.
The following additional packages will be installed:
  libc-dev-bin libc-devtools libc6-dev libcrypt-dev libnsl-dev libtirpc-dev
  linux-libc-dev manpages-dev rpcsvc-proto
```

注意: APT (Advanced Package Tool) 是用于查找和安装新包、升级包、清理包等的命令行工具...

`apt-get` 用于安装、更新软件包。它处理可用软件包的数据库。

我们使用的命令安装以下软件包:

混帐

跟踪并记录源代码开发过程中的所有更改。它还允许还原更改。

假根

制作假根环境的打包工具。

构建必备

three arguments:

```
$ tar [options] [archive-file]
[directory/file to be archived]
```

Xvf is the Unix-style, short method to implement `-extract -verbose -file`.

X = extracting the archive,

V = displaying verbose information

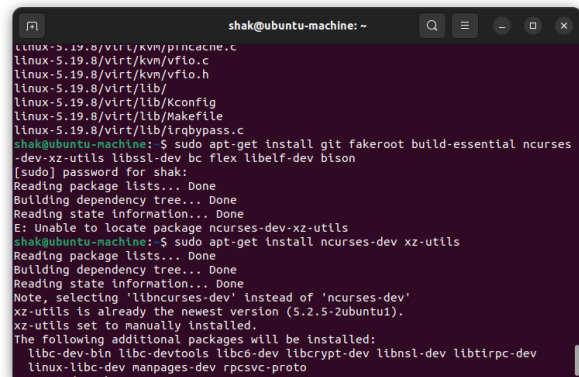
F = specifying a filename

Verbose information provides additional details about the task being carried out.

Step 3: Install Required Packages

In order to compile the kernel successfully, specific packages are required. To install these packages, we run the command:

```
Sudo apt-get install git fakeroot
build-essential ncurses-dev-xz-
utils libssl-dev bc flex libelf-
dev bison
```



```
shak@ubuntu-machine: ~
linux-5.19.8/virt/kvm/pnucacne.c
linux-5.19.8/virt/kvm/vfio.c
linux-5.19.8/virt/kvm/vfio.h
linux-5.19.8/virt/lib/
linux-5.19.8/virt/lib/Kconfig
linux-5.19.8/virt/lib/Makefile
linux-5.19.8/virt/lib/irqbypass.c
shak@ubuntu-machine: ~$ sudo apt-get install git fakeroot build-essential ncurses
-dev-xz-utils libssl-dev bc flex libelf-dev bison
[sudo] password for shak:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package ncurses-dev-xz-utils
shak@ubuntu-machine: ~$ sudo apt-get install ncurses-dev xz-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'libncurses-dev' instead of 'ncurses-dev'
xz-utils is already the newest version (5.2.5-2ubuntu1).
xz-utils set to manually installed.
The following additional packages will be installed:
  libc-dev-bin libc-devtools libc6-dev libcrypt-dev libnsl-dev libtirpc-dev
  linux-libc-dev manpages-dev rpcsvc-proto
```

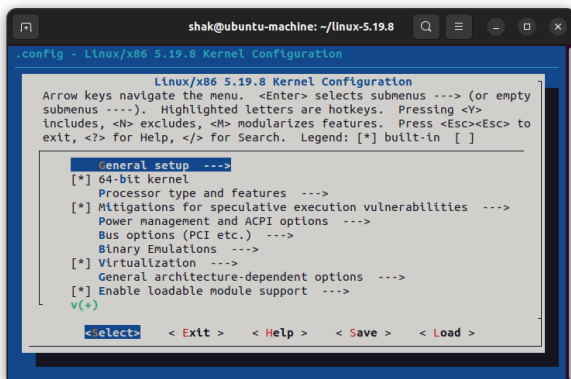
Note: APT (Advanced Package Tool) is the command line tool used to find and install new packages, upgrade packages, clean packages etc...

Apt-get is for installing, updating packages. It works on a database of available packages.

The command we used installs the following packages:

<p>安装开发工具，例如 C、C++、gcc 和 g++。</p> <p>ncurses-开发</p> <p>为基于文本的终端提供 API 的编程库。</p> <p>xz-utils</p> <p>提供快速的文件压缩和解压缩。</p> <p>libssl-开发</p> <p>支持加密数据并确保互联网连接安全的 SSL 和 TSL。</p> <p>bc（基本计算器）</p> <p>一种支持交互式执行语句的数学脚本语言。</p> <p>flex（快速词法分析器生成器）</p> <p>生成将字符转换为标记的词法分析器。</p> <p>自由开发</p> <p>发布用于管理 ELF 文件（可执行文件、核心转储和目标代码）的共享库</p> <p>野牛</p> <p>将语法描述转换为 C 程序的 GNU 解析器生成器。</p> <p>第 4 步：配置和编译</p> <p>在编译内核之前，我们需要配置哪些模块要包含，哪些模块要省略。</p> <p>有很多方法可以做到这一点。</p> <p>一种简单直接的方法是首先复制现有的内核配置文件，然后使用 “menuconfig”进行更改（如有必要）。这是最快的方法，也可能是最安全的方法。</p> <p>导航到 linux-5.9.6。使用 cd 命令的目录：</p> <pre>cd linux-5.9.6</pre> <pre>cp /boot/config-\$(uname -r) .config</pre> <p>Make menuconfig</p> <p>该命令启动几个脚本，然后打开配置菜单：</p>		
	Package	Package description
	git	Tracks and makes a record of all changes during development in the source code. It also allows reverting the changes.
	fakeroot	Packaging tool that makes the fake root environment.
	build-essential	Installs development tools such as C, C++, gcc, and g++.
	ncurses-dev	Programming library that provides API for the text-based terminals.
	xz-utils	Provides fast file compression and decompression.
	libssl-dev	Supports SSL and TSL that encrypt data and make the internet connection secure.
	bc (Basic Calculator)	A mathematical scripting language that supports the interactive execution of statements.
	flex (Fast Lexical Analyzer Generator)	Generates lexical analyzers that convert characters into tokens.
	libelf-dev	Issues a shared library for managing ELF files (executable files, core dumps and object code)
	bison	GNU parser generator

```
shak@ubuntu-machine: ~/linux-5.19.8
Setting up binutils-x86-64-linux-gnu (2.38-3ubuntu1) ...
Setting up binutils (2.38-3ubuntu1) ...
Setting up gcc-11 (11.2.0-19ubuntu1) ...
Setting up gcc (4:11.2.0-1ubuntu1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
shak@ubuntu-machine:~/linux-5.19.8$ make menuconfig
HOSTCC scripts/basic/fixdep
UPD scripts/kconfig/nconf.cfg
HOSTCC scripts/kconfig/nconf.o
HOSTCC scripts/kconfig/xdialog/checklist.o
HOSTCC scripts/kconfig/xdialog/inputbox.o
HOSTCC scripts/kconfig/xdialog/menubox.o
HOSTCC scripts/kconfig/xdialog/textbox.o
HOSTCC scripts/kconfig/xdialog/utill.o
HOSTCC scripts/kconfig/xdialog/yesno.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX scripts/kconfig/lexer.lex.c
/bin/sh: 1: flex: not found
make[1]: *** [scripts/Makefile.host:9: scripts/kconfig/lexer.lex.c] Error 127
make: *** [Makefile:629: menuconfig] Error 2
shak@ubuntu-machine:~/linux-5.19.8$ sudo apt-get install flex
Reading package lists... Done
```



that converts grammar description to a C program.

在这种情况下，我们不会对配置菜单进行任何更改，因此我们退出菜单。

注意：使用 `cp` 命令可以在不离开 Linux 终端的情况下执行复制和粘贴操作。该命令的语法是：

`cp [...文件/目录源][目标]`

`[file/directory-sources]` 指定要复制的文件或目录的来源

`[destination]` 参数指定要将文件复制到的位置。

`$(uname -r)` 用于命令替换。它运行 `uname -r` 命令，该命令返回当前内核版本，然后将返回的内容放入 `cp` 命令中。

命令替换允许命令的输出替换命令本身。当命令包含如下时，会发生命令替换：

`$ (命令)`

或者

`命令`

“make”命令可帮助系统管理员编译和安装开源实用程序。它执行或使用 makefile 中指定的参数来识别处理目标项目的不同操作。

第 5 步：更新 makefile 中的 EXTRAVERSION

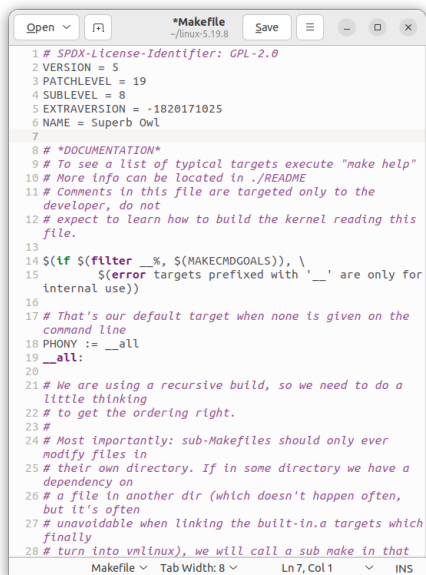
我们通过使用文本编辑器打开位于源代码根目录中的内核的 makefile 来做到这一点。

现在我们将 EXTRAVERSION 的值更改为 -1820171025。

-外向 =

+EXTRAVERSION = -1820171025

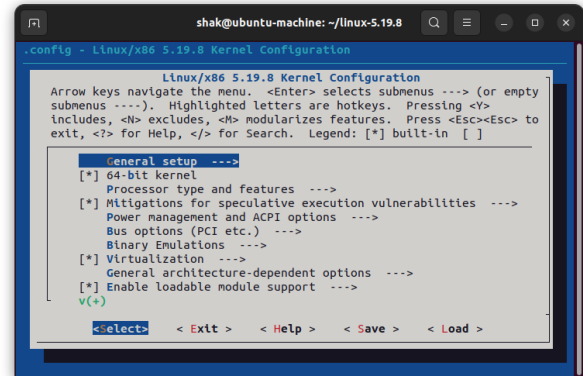
现在我们可以重新编译我们的内核



```
1 # SPDX-License-Identifier: GPL-2.0
2 VERSION = 5
3 PATCHLEVEL = 19
4 SUBLEVEL = 8
5 EXTRAVERSION = -1820171025
6 NAME = Superb Owl
7
8 # *DOCUMENTATION*
9 # To see a list of typical targets execute "make help"
10 # More info can be located in ./README
11 # Comments in this file are targeted only to the
12 # developer, do not
13 # expect to learn how to build the kernel reading this
14 # file.
15 $(if $(filter __%, $(MAKECMDGOALS)), \
16     $(error targets prefixed with '__' are only for
17     internal use))
18
19 # That's our default target when none is given on the
20 # command line
21 PHONY := __all
22 __all:
23
24 # We are using a recursive build, so we need to do a
25 # little thinking
26 # to get the ordering right.
27
28 # Most importantly: sub-Makefiles should only ever
29 # modify files in
30 # their own directory. If in some directory we have a
31 # dependency on
32 # a file in another dir (which doesn't happen often,
33 # but it's often
34 # unavoidable when linking the built-in.a targets which
35 # finally
36 # turn into vmlinux), we will call a sub make in that
```

第 6 步：构建内核

通过运行以下命令开始构建内核



In this case we won't be making any changes in the configuration menu so we exit the menu.

Note: One can perform copy and paste operations without leaving the Linux terminal using the `cp` command. The syntax of this command is:

```
cp [...file/directory-sources]
[destination]
```

[file/directory-sources] specifies the sources of the files or directories you want to copy

The [destination] argument specifies the location you want to copy the file to.

\$(uname -r) is used for command substitution. It runs the `uname -r` command which returns the current kernel version and then puts what is returned into the `cp` command.

Command substitution allows the output of a command to replace the command itself. Command substitution occurs when a command is enclosed as follows:

\$(command)

or

`command`

The ‘make’ command helps system administrators compile and install open-source

```
shak@ubuntu-machine: ~/linux-5.19.8
*** Execute 'make' to start the build or try 'make help'.

shak@ubuntu-machine:~/linux-5.19.8$ make
SYMC      include/config/auto.conf.cmd
HOSTCC    scripts/kconfig/conf.o
HOSTLD    scripts/kconfig/conf
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_la32.h
SYSHDR    arch/x86/include/generated/asm/unistd_64_x32.h
SYSTBL    arch/x86/include/generated/asm/syscalls_64.h
HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
HOSTCC    arch/x86/tools/relocs_32.o
HOSTCC    arch/x86/tools/relocs_64.o
HOSTCC    arch/x86/tools/relocs_common.o
HOSTLD    arch/x86/tools/relocs
HOSTCC    scripts/genksyms/genksyms.o
YACC      scripts/genksyms/parse.tab.[ch]
HOSTCC    scripts/genksyms/parse.tab.o
LEX       scripts/genksyms/lex.lex.c
HOSTCC    scripts/genksyms/lex.lex.o
```

构建和编译 Linux 内核的过程需要一些时间才能完成。

回答完一连串问题后，我们可以安装我们使用以下命令启用的模块：

utilities. It executes or uses the arguments specified in the makefile identifying different actions to handle the target project.

Step 5: update EXTRAVERSION in makefile

We do this by opening the kernel's makefile, located in the root directory in the source code using the text editor.

Now we change the value of EXTRAVERSION to -1820171025.

-EXTRAVERSION =

+EXTRAVERSION = -1820171025

Now we can re-compile our kernel

```
*Makefile
~/linux-5.19.8

1 # SPDX-License-Identifier: GPL-2.0
2 VERSION = 5
3 PATCHLEVEL = 19
4 SUBLEVEL = 8
5 EXTRAVERSION = -1820171025
6 NAME = Superb Owl
7
8 # *DOCUMENTATION*
9 # To see a list of typical targets execute "make help"
10 # More info can be located in ./README
11 # Comments in this file are targeted only to the
12 # developer, do not
13 # expect to learn how to build the kernel reading this
14 # file.
15
16 S(if $(filter __%, $(MAKECMDGOALS)), \
17   $(error targets prefixed with '___' are only for
18   internal use))
19
20 __all:
21
22 # We are using a recursive build, so we need to do a
23 # little thinking
24 # to get the ordering right.
25 #
26 # Most importantly: sub-Makefiles should only ever
27 # modify files in
28 # their own directory. If in some directory we have a
29 # dependency on
30 # a file in another dir (which doesn't happen often,
31 # but it's often
32 # unavoidable when linking the built-in.a targets which
33 # finally
34 # turn into vmlinux), we will call a sub make in that
35
36 Makefile Tab Width: 8 Ln 7, Col 1 INS
```

Step 6: Building the kernel

Start building the kernel by running the following command

The process of building and compiling the Linux kernel takes some time to complete.

sudo make modules_install

(这需要很多时间)

```
shak@ubuntu-machine: ~/linux-5.19.8
*** Execute 'make' to start the build or try 'make help'.

shak@ubuntu-machine:~/linux-5.19.8$ make
SYNC      include/config/auto.conf.cnd
HOSTCC    scripts/kconfig/conf.o
HOSTLD    scripts/kconfig/conf
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_la32.h
SYSHDR    arch/x86/include/generated/asm/unistd_64_x32.h
SYSTBL    arch/x86/include/generated/asm/syscalls_64.h
HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
HOSTCC    arch/x86/tools/relocs_32.o
HOSTCC    arch/x86/tools/relocs_64.o
HOSTCC    arch/x86/tools/relocs_common.o
HOSTLD    arch/x86/tools/relocs
HOSTCC    scripts/genksyms/genksyms.o
YACC      scripts/genksyms/parse.tab.[ch]
HOSTCC    scripts/genksyms/parse.tab.o
LEX       scripts/genksyms/lex.lex.c
HOSTCC    scripts/genksyms/lex.lex.o
```

After answering the litany of questions, we can then install the modules we've enabled with the command:

sudo make modules_install

(this takes a lot of time)

```
shak@ubuntu-machine: ~/linux-5.19.8
shak@shak-Satellite-C55-B: ~/linux-5.19.8

CC [M]    sound/virtio/virtio_snd.mod.o
LD [M]    sound/virtio/virtio_snd.ko
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/consts.py
shak@ubuntu-machine:~/linux-5.19.8$ sudo make modules_install
[sudo] password for shak:
INSTALL /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/aegis128-aesni.k
o
SIGN      /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/aegis128-aesni.k
o
INSTALL /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/aesni-intel.ko
SIGN      /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/aesni-intel.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/blowfish-x86_64.
ko
SIGN      /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/blowfish-x86_64.
ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/camellia-aesni-a
vx-x86_64.ko
SIGN      /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/camellia-aesni-a
vx-x86_64.ko
```

Finally, we install the kernel by typing

> sudo make install

```
shak@ubuntu-machine: ~/linux-5.19.8
shak@shakSatellite-C55-B: ~/linux-5.19.8
CC [M] sound/virtio/virtio_snd.mod.o
LD [M] sound/virtio/virtio_snd.ko
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/consts.py
shak@ubuntu-machine: ~/linux-5.19.8$ sudo make modules_install
[sudo] password for shak:
INSTALL /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/aegis128-aesni.k
o
SIGN /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/aegis128-aesni.k
o
INSTALL /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/aesni-intel.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/aesni-intel.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/blowfish-x86_64.
ko
SIGN /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/blowfish-x86_64.
ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/camellia-aesni-a
vx-x86_64.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/arch/x86/crypto/camellia-aesni-a
vx-x86_64.ko
```

```
shak@ubuntu-machine: ~/linux-5.19.8
shak@shakSatellite-C55-B: ~/linux-5.19.8
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/misc/snd-ua101.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/usb/snd-usb-audio.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/snd-usb-audio.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/usb/snd-usbmidi-lib.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/snd-usbmidi-lib.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/usb/usx2y/snd-usb-us122l.k
o
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/usx2y/snd-usb-us122l.k
o
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/virtio/virtio_snd.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/virtio/virtio_snd.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/x86/snd-hdmi-lpe-audio.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/x86/snd-hdmi-lpe-audio.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/xen/snd_xen_front.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/xen/snd_xen_front.ko
DEPMOD /lib/modules/5.19.8-1820171025
shak@ubuntu-machine: ~/linux-5.19.8$ sudo make install
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.19.8-1820171025 /b
oot/vmlinuz-5.19.8-1820171025
update-initramfs: Generating /boot/initrd.img-5.19.8-1820171025
```

(this takes a lot of time)

The output shows done when finished:

最后，我们通过键入来安装内核

> sudo make install

(这需要很多时间)

```
shak@ubuntu-machine: ~/linux-5.19.8
shak@shakSatellite-C55-B: ~/linux-5.19.8
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/misc/snd-ua101.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/usb/snd-usb-audio.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/snd-usb-audio.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/usb/snd-usbmidi-lib.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/snd-usbmidi-lib.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/usb/usx2y/snd-usb-us122l.k
o
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/usx2y/snd-usb-us122l.k
o
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/virtio/virtio_snd.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/virtio/virtio_snd.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/x86/snd-hdmi-lpe-audio.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/x86/snd-hdmi-lpe-audio.ko
INSTALL /lib/modules/5.19.8-1820171025/kernel/sound/xen/snd_xen_front.ko
SIGN /lib/modules/5.19.8-1820171025/kernel/sound/xen/snd_xen_front.ko
DEPMOD /lib/modules/5.19.8-1820171025
shak@ubuntu-machine: ~/linux-5.19.8$ sudo make install
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.19.8-1820171025 /b
oot/vmlinuz-5.19.8-1820171025
update-initramfs: Generating /boot/initrd.img-5.19.8-1820171025
```

```
shak@ubuntu-machine: ~/linux-5.19.8
shak@shakSatellite-C55-B: ~/linux-5.19.8
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.19.8-182017
1025 /boot/vmlinuz-5.19.8-1820171025
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-47-generic
I: /boot/initrd.img is now a symlink to initrd.img-5.19.8-1820171025
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.19.8-1820171025 /bo
ot/vmlinuz-5.19.8-1820171025
Sourcing file /etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.19.8-1820171025
Found initrd image: /boot/initrd.img-5.19.8-1820171025
Found linux image: /boot/vmlinuz-5.15.0-47-generic
Found initrd image: /boot/initrd.img-5.15.0-47-generic
Found linux image: /boot/vmlinuz-5.15.0-43-generic
Found initrd image: /boot/initrd.img-5.15.0-43-generic
Mentest86+ needs a 16-bit boot, that is not available on EFI, exiting
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot
entries.
Found Windows Boot Manager on /dev/sda1/EFI/Microsoft/Boot/bootmgfw.efi
Found Linux Mint 21 Vanessa (21) on /dev/sdas
Adding boot menu entry for UEFI Firmware Settings ...
done
shak@ubuntu-machine: ~/linux-5.19.8$
```

Step 7: Reboot and verify kernel version

Now we reboot the machine.

When the system boots up, verify the kernel version using the uname command:

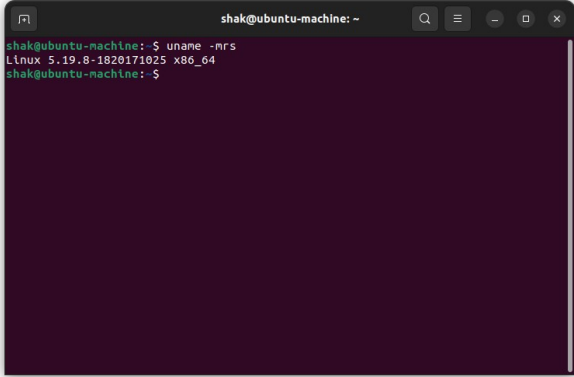
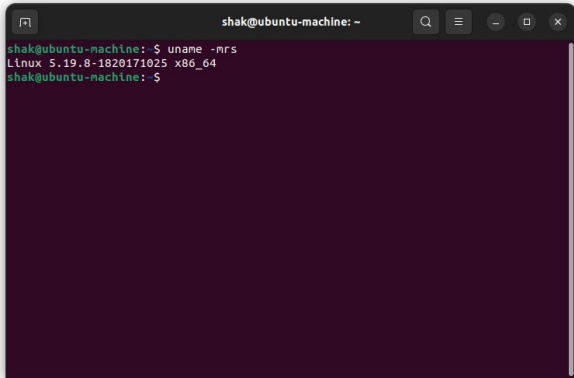
Uname -mrs

The terminal prints out the current Linux kernel version.

完成后输出显示完成:

```
shak@ubuntu-machine: ~/linux-5.19.8
shak@shakSatellite-C55-B: ~/linux-5.19.8
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.19.8-182017
1025 /boot/vmlinuz-5.19.8-1820171025
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-47-generic
I: /boot/initrd.img is now a symlink to initrd.img-5.19.8-1820171025
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.19.8-1820171025 /bo
ot/vmlinuz-5.19.8-1820171025
Sourcing file /etc/default/grub'
Sourcing file /etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.19.8-1820171025
Found initrd image: /boot/initrd.img-5.19.8-1820171025
Found linux image: /boot/vmlinuz-5.15.0-47-generic
Found initrd image: /boot/initrd.img-5.15.0-47-generic
Found linux image: /boot/vmlinuz-5.15.0-43-generic
Found initrd image: /boot/initrd.img-5.15.0-43-generic
Mentest86+ needs a 16-bit boot, that is not available on EFI, exiting
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot
entries.
Found Windows Boot Manager on /dev/sda1/EFI/Microsoft/Boot/bootmgfw.efi
Found Linux Mint 21 Vanessa (21) on /dev/sdas
Adding boot menu entry for UEFI Firmware Settings ...
done
shak@ubuntu-machine: ~/linux-5.19.8$
```

```
shak@ubuntu-machine: ~
shak@ubuntu-machine: $ uname -mrs
Linux 5.19.8-1820171025 x86_64
shak@ubuntu-machine: $
```

实验结果和分析	Experimental Results and Analysis
<p>自定义内核编译成功，学号成功添加到 makefile 中的 EXTRAVERSION。</p> <p>重启后的 <code>uname -mrs</code> 命令显示结果</p>  <pre>shak@ubuntu-machine: ~ shak@ubuntu-machine:~\$ uname -mrs Linux 5.19.8-1820171025 x86_64 shak@ubuntu-machine:~\$</pre>	<p>The custom kernel was successfully compiled and student number was successfully added to EXTRAVERSION in the makefile.</p> <p>The <code>uname -mrs</code> command after rebooting shows us the results</p>  <pre>shak@ubuntu-machine: ~ shak@ubuntu-machine:~\$ uname -mrs Linux 5.19.8-1820171025 x86_64 shak@ubuntu-machine:~\$</pre>

讨论、心得	Discussion and experience
<p>确定执行此实验的最佳步骤需要一些时间。即便如此，最初的方法还是遗漏了一些东西。</p> <p>我需要安装 gcc、flex、bison 才能使 make menuconfig 命令正常工作。</p>	<p>It took some time to identify the best steps to execute this experiment. Even then there were some things that were missed in the initial approach.</p> <p>I needed to install gcc, flex, bison to make the make menuconfig command work.</p>

```

shak@ubuntu-machine: ~/linux-5.19.8
/bin/sh: 1: gcc: not found
make[1]: *** [scripts/Makefile.host:95: scripts/basic/fixdep] Error 127
make: *** [Makefile:565: scripts_basic] Error 2
shak@ubuntu-machine: ~/linux-5.19.8$ cp -v /boot/config .config
cp: cannot stat '/boot/config': No such file or directory
shak@ubuntu-machine: ~/linux-5.19.8$ cp -v /boot/config- .config
cp: cannot stat '/boot/config-': No such file or directory
shak@ubuntu-machine: ~/linux-5.19.8$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-47-generic' -> '.config'
shak@ubuntu-machine: ~/linux-5.19.8$ make menuconfig
HOSTCC scripts/basic/fixdep
/bin/sh: 1: gcc: not found
make[1]: *** [scripts/Makefile.host:95: scripts/basic/fixdep] Error 127
make: *** [Makefile:565: scripts_basic] Error 2
shak@ubuntu-machine: ~/linux-5.19.8$ sudo apt-get install gcc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu gcc-11 libasan6
  libatomic1 libbinutils libbtf-nobfd libbtf-nobfd libgcc-11-dev libitm1
  liblsan0 libquadmath0 libtsan0 libubsan1
Suggested packages:
  binutils-doc gcc-multilib autoconf automake libtool flex bison gcc-doc

```

运行“make”命令时。为了解决这个问题，我必须安装 libssl-dev 和 libelf-dev。

```

shak@ubuntu-machine: ~/linux-5.19.8
HOSTCC arch/x86/tools/relocs_64.o
HOSTCC arch/x86/tools/relocs_common.o
HOSTLD arch/x86/tools/relocs
HOSTCC scripts/genksyms/genksyms.o
YACC scripts/genksyms/parse.tab.[ch]
HOSTCC scripts/genksyms/parse.tab.o
LEX scripts/genksyms/lex.lex.c
HOSTCC scripts/genksyms/lex.lex.o
HOSTLD scripts/genksyms/genksyms
HOSTCC scripts/selinux/genheaders/genheaders
HOSTCC scripts/selinux/ndp/ndp
HOSTCC scripts/bin2c
HOSTCC scripts/kallsyms
HOSTCC scripts/sorttable
HOSTCC scripts/asn1_compiler
HOSTCC scripts/sign-file
scripts/sign-file.c:25:10: fatal error: openssl/opensslv.h: No such file or directory
   25 | #include <openssl/opensslv.h>
      |
compilation terminated.
make[1]: *** [scripts/Makefile.host:95: scripts/sign-file] Error 1
make: *** [Makefile:1188: scripts] Error 2
shak@ubuntu-machine: ~/linux-5.19.8$

```

我遇到了这些错误，因为我的第一个 sudo apt-get install 命令有一个语法错误，省略了某些软件包的安装。

```

shak@ubuntu-machine: ~
linux-5.19.8/virt/kvm/princacne.c
linux-5.19.8/virt/kvm/vfio.c
linux-5.19.8/virt/kvm/vfio.h
linux-5.19.8/virt/lib/
linux-5.19.8/virt/lib/Kconfig
linux-5.19.8/virt/lib/Makefile
linux-5.19.8/virt/lib/irqbypass.c
shak@ubuntu-machine: $ sudo apt-get install git fakeroot build-essential ncurses
-dev-xz-utils libssl-dev bc flex libelf-dev bison
[sudo] password for shak:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package ncurses-dev-xz-utils
shak@ubuntu-machine: $ sudo apt-get install ncurses-dev xz-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'libncurses-dev' instead of 'ncurses-dev'
xz-utils is already the newest version (5.2.5-2ubuntu1).
xz-utils set to manually installed.
The following additional packages will be installed:
  libc-dev-bin libc-devtools libc6-dev libcrypt-dev libnsl-dev libtirpc-dev
  linux-libc-dev manpages-dev rpcsvc-proto

```

在运行 make 命令时，我遇到了认证错误。要修复它，我必须更改我的配置文件 .config

```

shak@ubuntu-machine: ~/linux-5.19.8
/bin/sh: 1: gcc: not found
make[1]: *** [scripts/Makefile.host:95: scripts/basic/fixdep] Error 127
make: *** [Makefile:565: scripts_basic] Error 2
shak@ubuntu-machine: ~/linux-5.19.8$ cp -v /boot/config .config
cp: cannot stat '/boot/config': No such file or directory
shak@ubuntu-machine: ~/linux-5.19.8$ cp -v /boot/config- .config
cp: cannot stat '/boot/config-': No such file or directory
shak@ubuntu-machine: ~/linux-5.19.8$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-47-generic' -> '.config'
shak@ubuntu-machine: ~/linux-5.19.8$ make menuconfig
HOSTCC scripts/basic/fixdep
/bin/sh: 1: gcc: not found
make[1]: *** [scripts/Makefile.host:95: scripts/basic/fixdep] Error 127
make: *** [Makefile:565: scripts_basic] Error 2
shak@ubuntu-machine: ~/linux-5.19.8$ sudo apt-get install gcc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu gcc-11 libasan6
  libatomic1 libbinutils libbtf-nobfd libbtf-nobfd libgcc-11-dev libitm1
  liblsan0 libquadmath0 libtsan0 libubsan1
Suggested packages:
  binutils-doc gcc-multilib autoconf automake libtool flex bison gcc-doc

```

When running the ‘make’ command. To fix this I had to install libssl-dev and libelf-dev.

```

shak@ubuntu-machine: ~/linux-5.19.8
HOSTCC arch/x86/tools/relocs_64.o
HOSTCC arch/x86/tools/relocs_common.o
HOSTLD arch/x86/tools/relocs
HOSTCC scripts/genksyms/genksyms.o
YACC scripts/genksyms/parse.tab.[ch]
HOSTCC scripts/genksyms/parse.tab.o
LEX scripts/genksyms/lex.lex.c
HOSTCC scripts/genksyms/lex.lex.o
HOSTLD scripts/genksyms/genksyms
HOSTCC scripts/selinux/genheaders/genheaders
HOSTCC scripts/selinux/ndp/ndp
HOSTCC scripts/bin2c
HOSTCC scripts/kallsyms
HOSTCC scripts/sorttable
HOSTCC scripts/asn1_compiler
HOSTCC scripts/sign-file
scripts/sign-file.c:25:10: fatal error: openssl/opensslv.h: No such file or directory
   25 | #include <openssl/opensslv.h>
      |
compilation terminated.
make[1]: *** [scripts/Makefile.host:95: scripts/sign-file] Error 1
make: *** [Makefile:1188: scripts] Error 2
shak@ubuntu-machine: ~/linux-5.19.8$

```

I ran into these errors because my first sudo apt-get install command had a syntax error that omitted installation of some packages.

```

shak@ubuntu-machine: ~
linux-5.19.8/virt/kvm/princacne.c
linux-5.19.8/virt/kvm/vfio.c
linux-5.19.8/virt/kvm/vfio.h
linux-5.19.8/virt/lib/
linux-5.19.8/virt/lib/Kconfig
linux-5.19.8/virt/lib/Makefile
linux-5.19.8/virt/lib/irqbypass.c
shak@ubuntu-machine: $ sudo apt-get install git fakeroot build-essential ncurses
-dev-xz-utils libssl-dev bc flex libelf-dev bison
[sudo] password for shak:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package ncurses-dev-xz-utils
shak@ubuntu-machine: $ sudo apt-get install ncurses-dev xz-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'libncurses-dev' instead of 'ncurses-dev'
xz-utils is already the newest version (5.2.5-2ubuntu1).
xz-utils set to manually installed.
The following additional packages will be installed:
  libc-dev-bin libc-devtools libc6-dev libcrypt-dev libnsl-dev libtirpc-dev
  linux-libc-dev manpages-dev rpcsvc-proto

```

While running the make command I ran into a certification error. To fix it I had to change my config file .config

CONFIG_SYSTEM_TRUSTED_KEYS="debian/canonical-certs.pem"

至

CONFIG_SYSTEM_TRUSTED_KEYS=""

A screenshot of a terminal window titled "linux-5.19.8" showing the configuration of CONFIG_SYSTEM_TRUSTED_KEYS. The terminal output shows the configuration of CONFIG_SYSTEM_TRUSTED_KEYS as an empty string. The terminal window has a menu bar with "Open", "Save", and "Close" buttons. The status bar at the bottom shows "Plain Text", "Tab Width: 8", "Ln 11102, Col 32", and "INS".

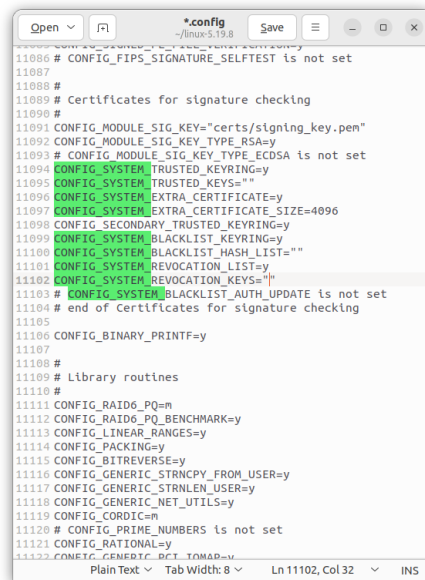
```
11086 # CONFIG_FIPS_SIGNATURE_SELFTEST is not set
11087
11088 #
11089 # Certificates for signature checking
11090 #
11091 CONFIG_MODULE_SIG_KEY="certs/signing_key.pem"
11092 CONFIG_MODULE_SIG_KEY_TYPE_RSA=y
11093 # CONFIG_MODULE_SIG_KEY_TYPE_ECDSA is not set
11094 CONFIG_SYSTEM_TRUSTED_KEYRING=y
11095 CONFIG_SYSTEM_TRUSTED_KEYS=""
11096 CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
11097 CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
11098 CONFIG_SECONDARY_TRUSTED_KEYRING=y
11099 CONFIG_SYSTEM_BLACKLIST_KEYRING=y
11100 CONFIG_SYSTEM_BLACKLIST_HASH_LIST=""
11101 CONFIG_SYSTEM_REVOCATION_LIST=y
11102 CONFIG_SYSTEM_REVOCATION_KEYS=""
11103 # CONFIG_SYSTEM_BLACKLIST_AUTH_UPDATE is not set
11104 # end of Certificates for signature checking
11105
11106 CONFIG_BINARY_PRINTF=y
11107
11108 #
11109 # Library routines
11110 #
11111 CONFIG_RAID6_PQ=m
11112 CONFIG_RAID6_PQ_BENCHMARK=y
11113 CONFIG_LINEAR_RANGES=y
11114 CONFIG_PACKING=y
11115 CONFIG_BITREVERSE=y
11116 CONFIG_GENERIC_STRNCPY_FROM_USER=y
11117 CONFIG_GENERIC_STRNLEN_USER=y
11118 CONFIG_GENERIC_NET_UTILS=y
11119 CONFIG_CORDIC=m
11120 # CONFIG_PRIME_NUMBERS is not set
11121 CONFIG_RATIONAL=y
11122 CONFIG_GENERIC_DCT_TOMAP=y
11123 CONFIG_GENERIC_DCT_TOMAP_U
```

这一切都需要几个小时才能完成

CONFIG_SYSTEM_TRUSTED_KEYS="debian/canonical-certs.pem"

to

CONFIG_SYSTEM_TRUSTED_KEYS=""

A screenshot of a terminal window titled "linux-5.19.8" showing the configuration of CONFIG_SYSTEM_TRUSTED_KEYS. The terminal output shows the configuration of CONFIG_SYSTEM_TRUSTED_KEYS as an empty string. The terminal window has a menu bar with "Open", "Save", and "Close" buttons. The status bar at the bottom shows "Plain Text", "Tab Width: 8", "Ln 11102, Col 32", and "INS".

```
11086 # CONFIG_FIPS_SIGNATURE_SELFTEST is not set
11087
11088 #
11089 # Certificates for signature checking
11090 #
11091 CONFIG_MODULE_SIG_KEY="certs/signing_key.pem"
11092 CONFIG_MODULE_SIG_KEY_TYPE_RSA=y
11093 # CONFIG_MODULE_SIG_KEY_TYPE_ECDSA is not set
11094 CONFIG_SYSTEM_TRUSTED_KEYRING=y
11095 CONFIG_SYSTEM_TRUSTED_KEYS=""
11096 CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
11097 CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
11098 CONFIG_SECONDARY_TRUSTED_KEYRING=y
11099 CONFIG_SYSTEM_BLACKLIST_KEYRING=y
11100 CONFIG_SYSTEM_BLACKLIST_HASH_LIST=""
11101 CONFIG_SYSTEM_REVOCATION_LIST=y
11102 CONFIG_SYSTEM_REVOCATION_KEYS=""
11103 # CONFIG_SYSTEM_BLACKLIST_AUTH_UPDATE is not set
11104 # end of Certificates for signature checking
11105
11106 CONFIG_BINARY_PRINTF=y
11107
11108 #
11109 # Library routines
11110 #
11111 CONFIG_RAID6_PQ=m
11112 CONFIG_RAID6_PQ_BENCHMARK=y
11113 CONFIG_LINEAR_RANGES=y
11114 CONFIG_PACKING=y
11115 CONFIG_BITREVERSE=y
11116 CONFIG_GENERIC_STRNCPY_FROM_USER=y
11117 CONFIG_GENERIC_STRNLEN_USER=y
11118 CONFIG_GENERIC_NET_UTILS=y
11119 CONFIG_CORDIC=m
11120 # CONFIG_PRIME_NUMBERS is not set
11121 CONFIG_RATIONAL=y
11122 CONFIG_GENERIC_DCT_TOMAP=y
11123 CONFIG_GENERIC_DCT_TOMAP_U
```

This all took several hours to complete