**实验三 Linux内核编译**

实验日期：9月27日 地点：求是楼207

1. **实验目的**

本实验旨在通过亲手操作Linux内核的编译过程，深入理解和掌握Linux操作系统的核心构建机制，同时根据实际需求优化内核配置，提升系统性能，并在此过程中培养解决实际编译问题的能力。

1. **实验内容**
2. 下载并解压Linux内核源代码到指定目录，安装必要的编译工具和依赖库，如gcc、make、libncurses-dev等。
3. 进入内核源代码目录，执行make menuconfig命令，进入内核配置界面。保存配置文件，生成.config文件，该文件包含了内核编译的配置信息。
4. 执行make命令，编译内核。编译过程可能需要一段时间，具体时间取决于电脑性能和内核配置。编译过程中，系统会生成多个目标文件和可执行文件，最终生成新的内核映像文件。
5. 编译完成后，执行make modules\_install命令，将编译好的内核模块安装到系统中。执行make install命令，将新的内核映像文件安装到系统的启动目录中。
6. 重启系统，在启动菜单中选择新编译的内核进行启动。进入系统后，使用uname -r命令查看当前运行的内核版本，确认新内核已成功启动。
7. **实验过程记录**
8. 准备编译环境：下载gcc编译器，make项目构建工具

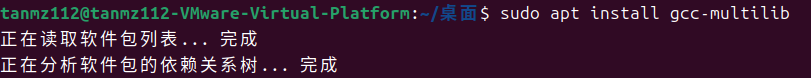


图 3- 1 安装gcc编译器



图 3- 2 安装make构建工具

1. 从Linux官方网站（[www.kernel.org](http://www.kernel.org/" \t "https://yiyan.baidu.com/chat/_blank)）下载内核源代码，为了实验方便，我选择一个相对较小的内核Linux5.15.1

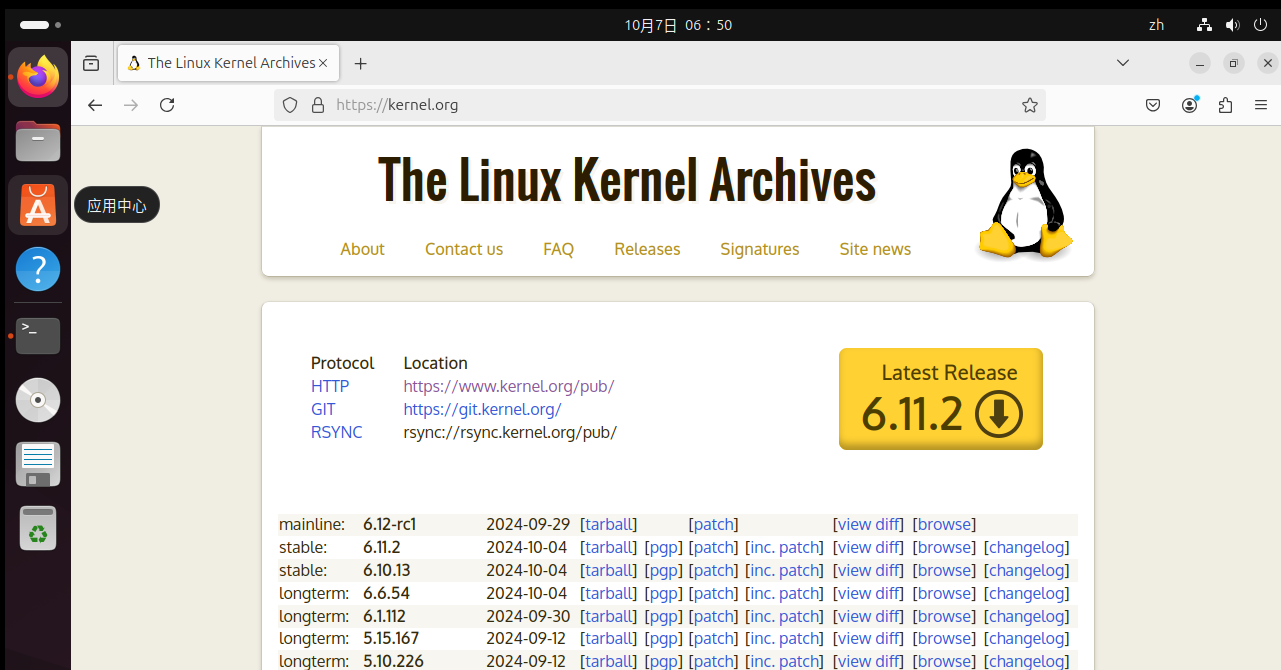


图 3- 3 Linux官网

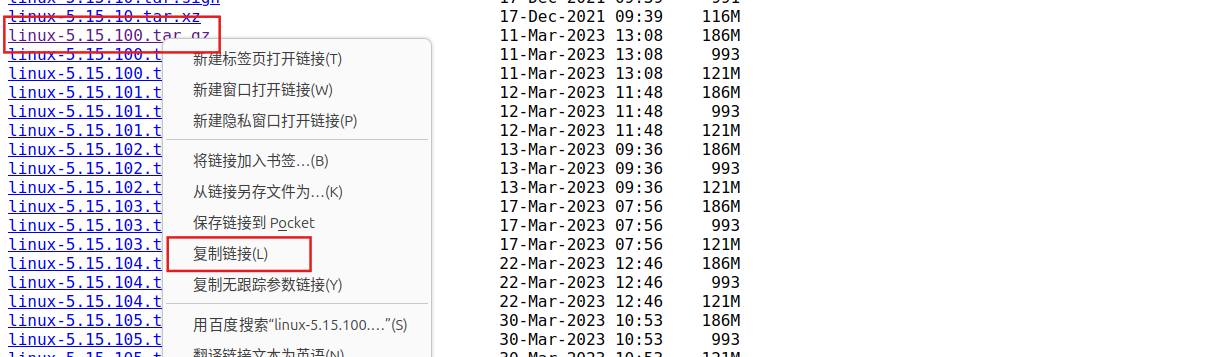


图 3- 4 下载Linux-5.15.100.tar.gz



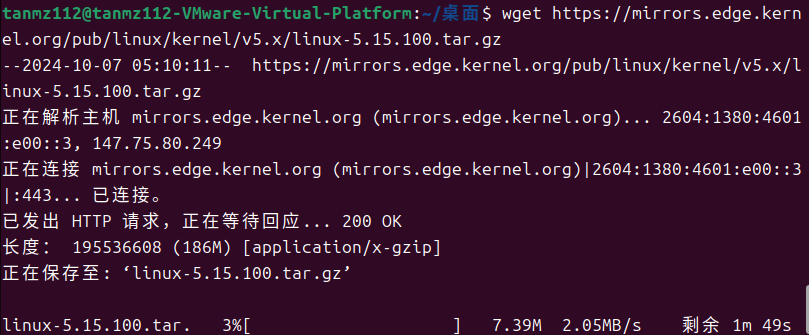
图 3- 5 用wget命令下载到桌面

图 3- 6 下载完成

1. 进入解压后的内核源代码目录。执行make menuconfig命令，进入内核配置界面。在这里，可以根据实际需求选择编译哪些内核模块和驱动程序。保存配置文件，生成.config文件，该文件包含了内核编译的配置信息。



图 3- 7 进入内核文件夹

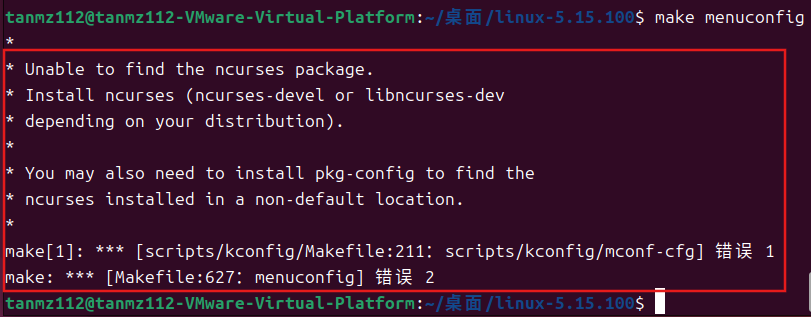


图 3- 8 保存配置文件出现错误



图 3- 9 安装依赖

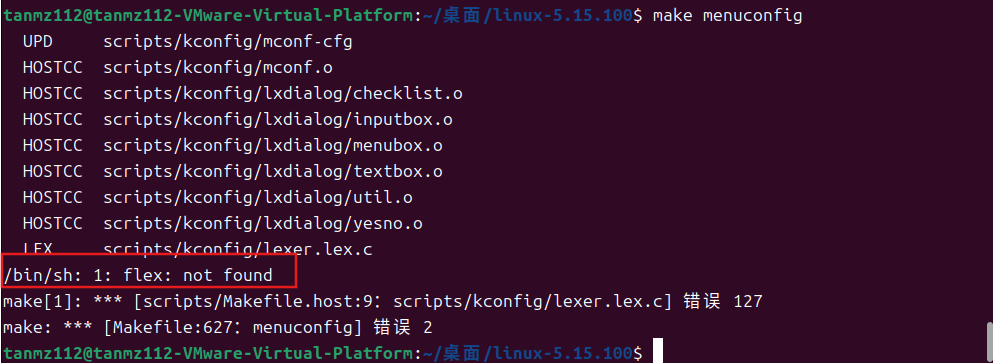


图 3- 10 发生错误



图 3- 11 安装依赖

1. 根据前辈的经验和我自己解决实验时报错问题得在编译前，首先要安装以下几种依赖：

|  |
| --- |
| sudo apt-get install gcc make libncurses5-dev openssl libssl-dev  sudo apt-get install build-essential  sudo apt-get install [pkg-config](https://zhida.zhihu.com/search?content_id=232178042&content_type=Article&match_order=1&q=pkg-config&zhida_source=entity" \t "https://zhuanlan.zhihu.com/p/_blank)  sudo apt-get install libc6-dev  sudo apt-get install bison  sudo apt-get install flex  sudo apt-get install libelf-dev  sudo apt-get install dwarves |

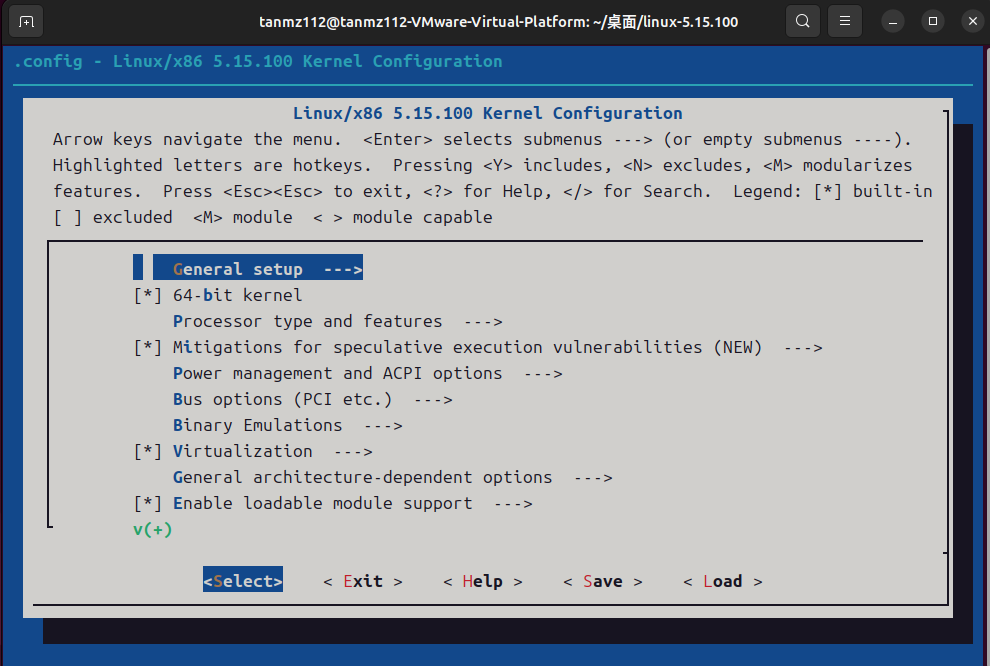


图 3- 12 配置文件图形界面



图 3- 13 进入配置文件

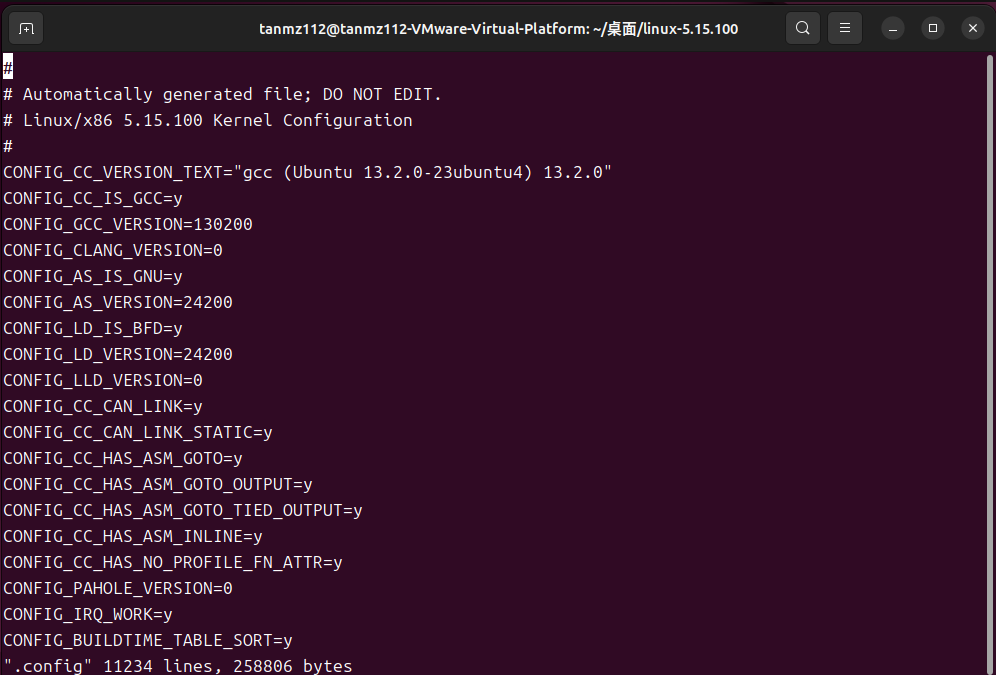


图 3- 14 配置文件原始界面

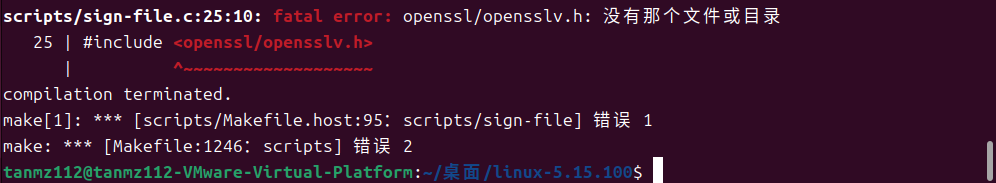


图 3- 15 缺少依赖的错误



图 3- 16 安装对应依赖

1. 使用make -jN命令开始编译内核，其中N是希望并行编译的任务数（通常设置为CPU核心数的两倍左右）。编译过程需要一段时间，具体时间取决于系统性能和内核源码的复杂性。



图 3- 17 编译内核



图 3- 18 缺少依赖报错



图 3- 19 安装依赖

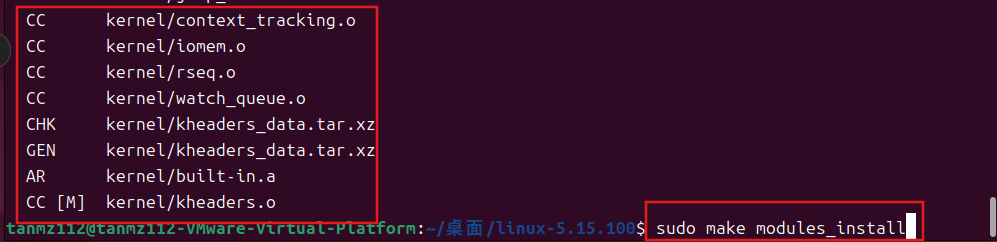


图 3- 20 编译失败

1. 在编译前，使用make clean命令清理之前的编译缓存和临时文件。用make -jN命令开始编译内核使用sudo make modules\_install命令将编译好的模块安装到系统的模块目录中。执行make install命令，将新的内核映像文件安装到系统的启动目录中。

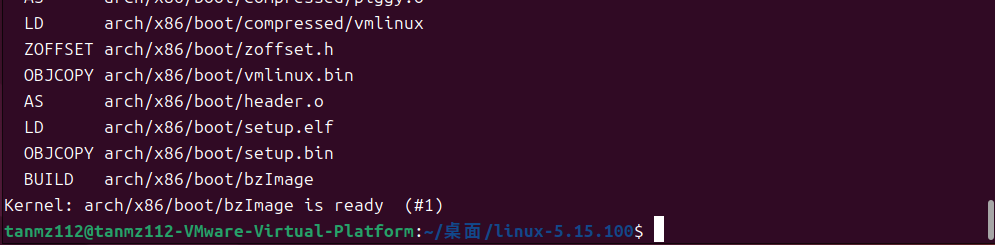


图 3- 21 编译成功

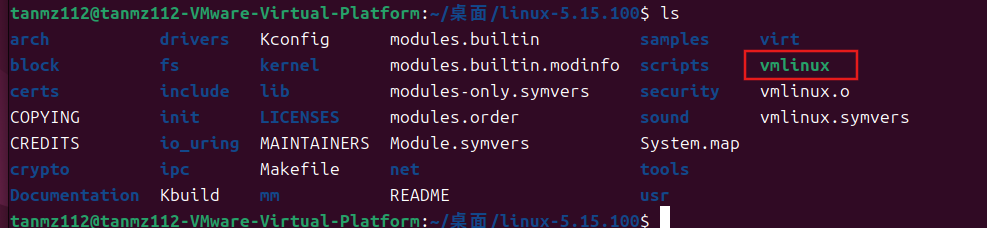


图 3- 22 查看编译后的文件

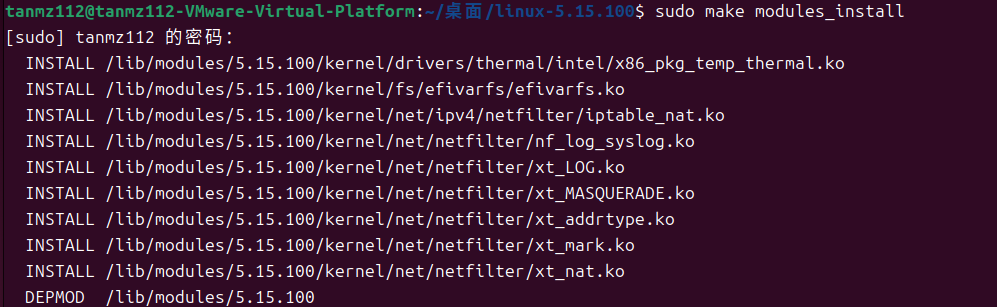


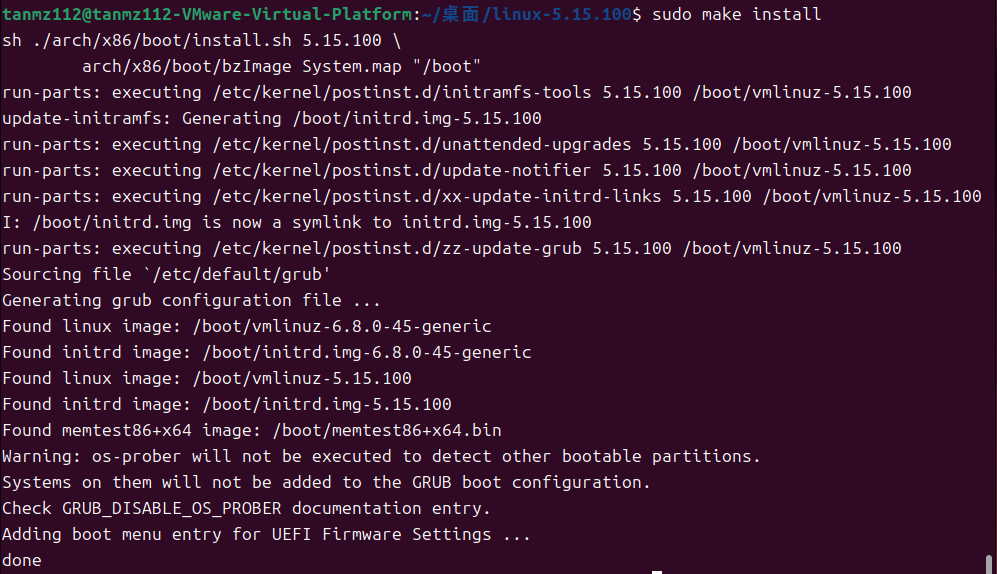
图 3- 23 安装内核模块

图 3- 24 安装到系统启动目录

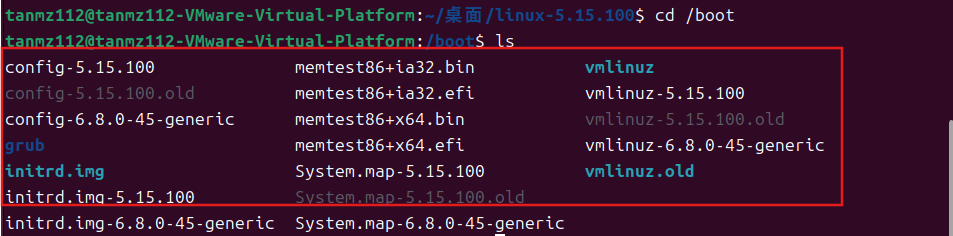


图 3- 25 查看启动目录

1. 编辑系统的GRUB配置文件（/boot/grub/grub.cfg），修改原文件：
   1. 注释 set timeout\_style = hidden
   2. 修改timeout的值为10
2. 将新内核添加到引导菜单中通过注释掉这一行（在行首添加 #），实际上是在禁用这个隐藏菜单的功能，使得GRUB菜单在启动时始终可见，无论是否按下任何键。将 timeout 设置为 10 意味GRUB将在启动时显示菜单，并等待10秒钟让用户做出选择。如果用户在这10秒内没有做出任何选择，GRUB将自动选择并启动默认的内核。

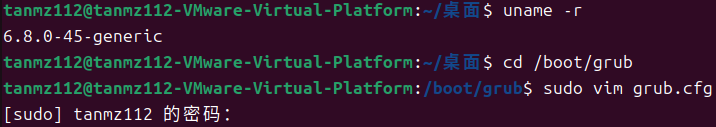


图 3- 26 进入grub程序

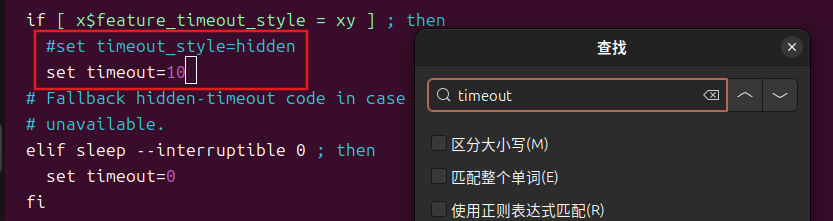


图 3- 27 查找并修改

1. 重启系统以加载新编译的内核。使用uname -r命令检查当前运行的内核版本，确认新编译的内核已正确加载。

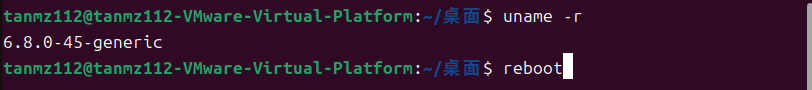


图 3- 28 查看当前内核和重启



图 3- 29 重启界面



图 3- 30 选择linux-5.15.100

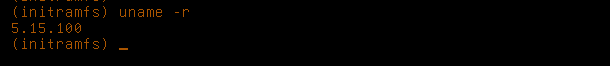


图 3- 31 切换内核成功

1. **实验小结**

通过本次实验，我深刻理解了Linux操作系统的核心构建机制，掌握了Linux内核编译的完整流程。在实验过程中，我不仅学会了如何安装编译工具和依赖库、如何配置内核、如何编译和安装内核模块及映像文件，还学会了如何修改GRUB配置文件以确保新编译的内核能够被正确加载。

在编译Linux内核时，如果遇到编译失败的情况，可以按照以下步骤进行排查和解决：

1. 检查编译环境
   1. 确认编译工具已安装：确保系统上安装了必要的编译工具，如GCC、Make等。检查这些工具的版本是否与内核源码兼容。
   2. 检查依赖库：编译内核可能需要一些特定的依赖库，如libncurses库等。使用包管理器（如apt-get、yum等）安装缺失的依赖库。
   3. 清理编译缓存：在编译前，使用make clean命令清理之前的编译缓存和临时文件。

这有助于避免由于旧文件导致的编译错误。

1. 检查内核配置文件
   1. 重新配置内核：使用make menuconfig或其他配置工具重新配置内核选项。确保所有必要的选项都已正确设置，特别是与硬件相关的选项。
   2. 检查配置文件完整性：确保配置文件没有损坏或缺失。可以尝试使用默认配置文件作为起点，然后逐步添加所需的选项。
2. 检查内核源码
   1. 确认源码完整性：从Linux内核官方网站下载内核源码。使用git status等命令检查源码状态，确保没有遗漏或错误的提交。
   2. 确保内核源码在编译前没有被不当修改。如果怀疑源码被修改，可以尝试重新下载并解压源码。
3. 检查编译日志：仔细阅读编译过程中生成的日志信息。日志中通常会包含导致编译错误的信息。