

（深圳）

实验报告

开课学期： 2024秋季

课程名称： 操作系统

实验名称：基于FUSE的青春版EXT2文件系统

学生班级： 5

学生学号： 220110515

学生姓名： 金正达

评阅教师：

报告成绩：

实验与创新实践教育中心制

2024年9月

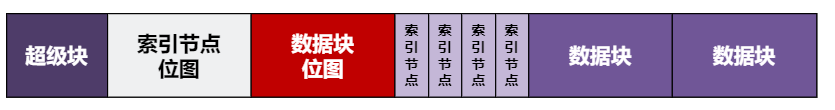
# 一、实验详细设计

*图文并茂地描述实验实现的所有功能和详细的设计方案及实验过程中的特色部分。*

1. **总体设计方案**

*详细阐述文件系统的总体设计思路，包括系统架构图和关键组件的说明。*

文件系统结构如下：



该简易ext2文件系统布局为：

超级块 1逻辑块，索引节点位图 1逻辑块，数据位图 1逻辑块，索引节点 12逻辑块，数据区 4081逻辑块。

一个逻辑块为两个IO块的大小，即1024B。

在此基础上，实现了文件系统的挂载、卸载、创建文件和目录、显示文件和目录的功能。

1. **功能详细说明**

*每个功能点的详细说明（关键的数据结构、核心代码、流程等）*

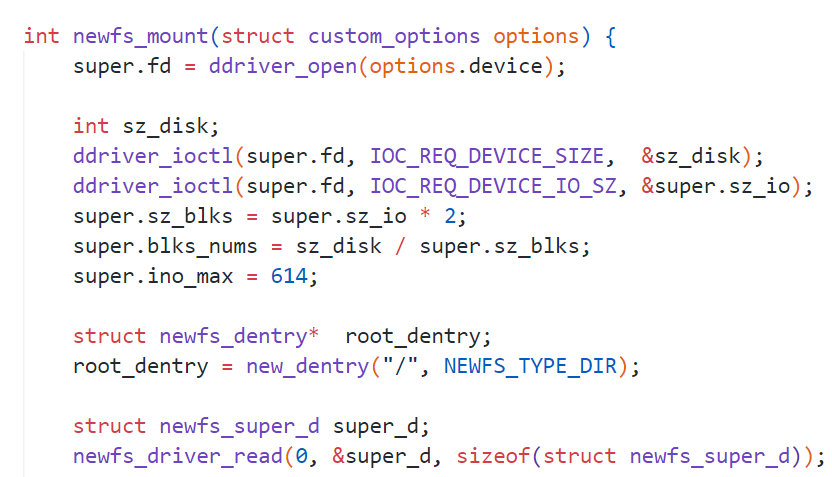
1. 挂载



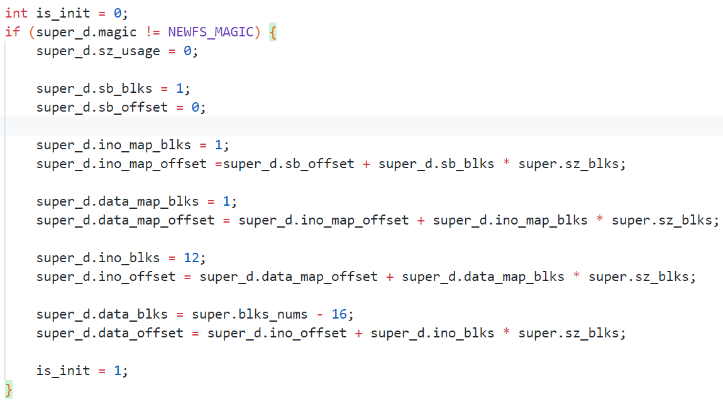
挂载过程直接调用了辅助函数进行挂载。

辅助函数中，首先，

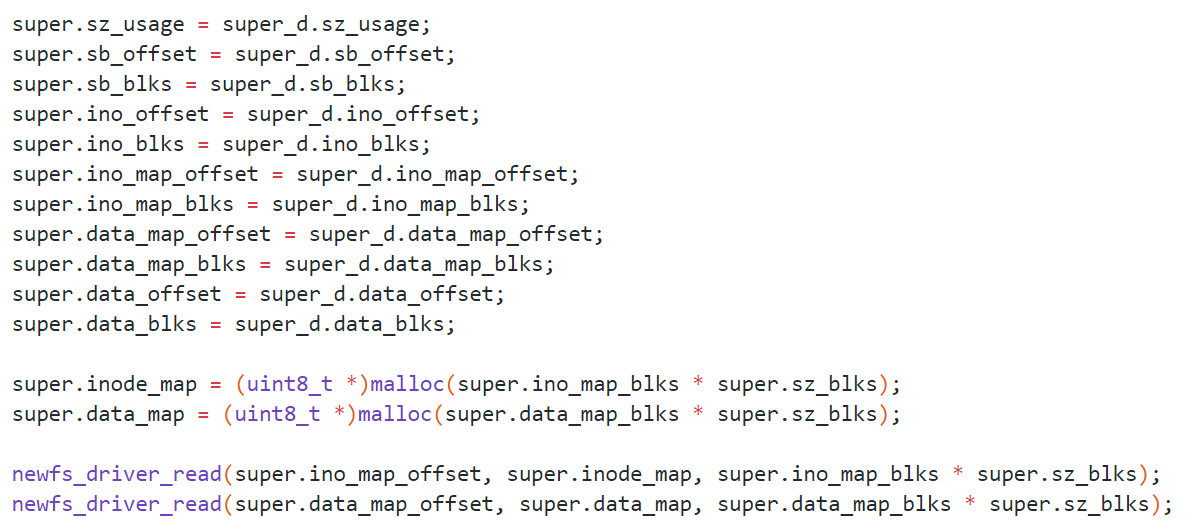
获取基本的io信息，并创建新的根目录，读取存储在磁盘中的超级块：



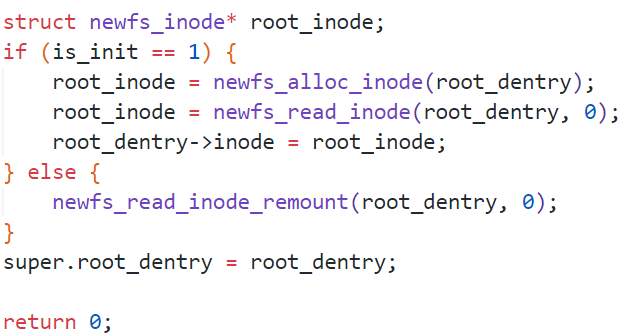
接着，判断文件系统是否已经初始化，若没有初始化，则填充超级块、索引位图块、数据位图块、索引节点块以及数据块的位置和大小，并标记进行了初始化：



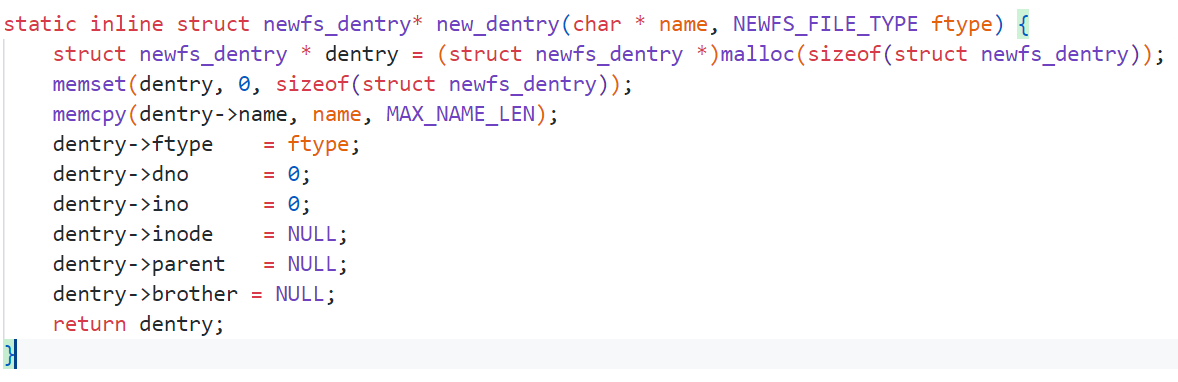
然后，将磁盘超级块中的信息填入内存超级块中，并创建读取索引和数据位图：



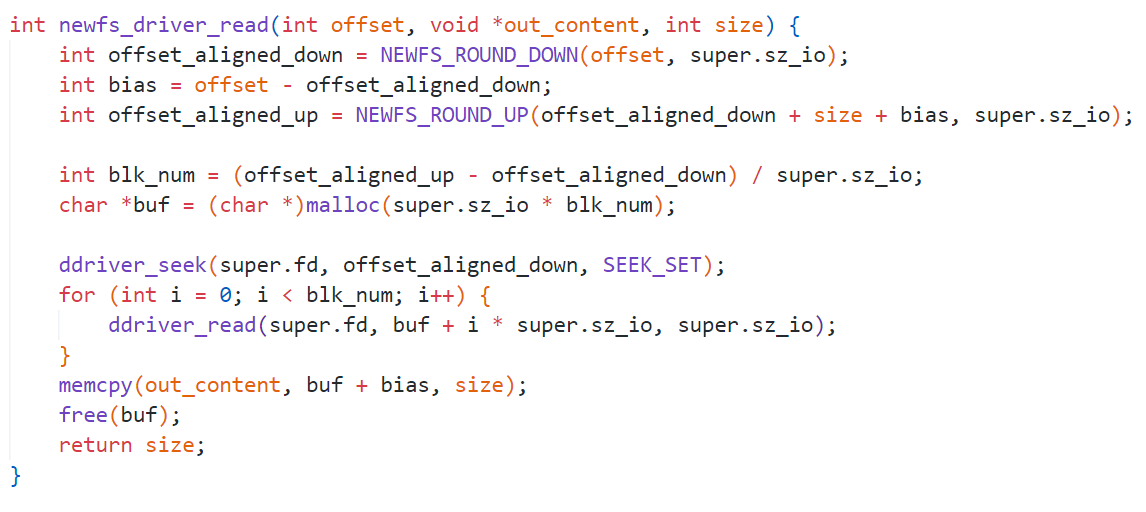
最后，如果此次挂载进行了初始化，就为根目录分配新的inode，否则，读取磁盘中的文件的层级结构，完成挂载：



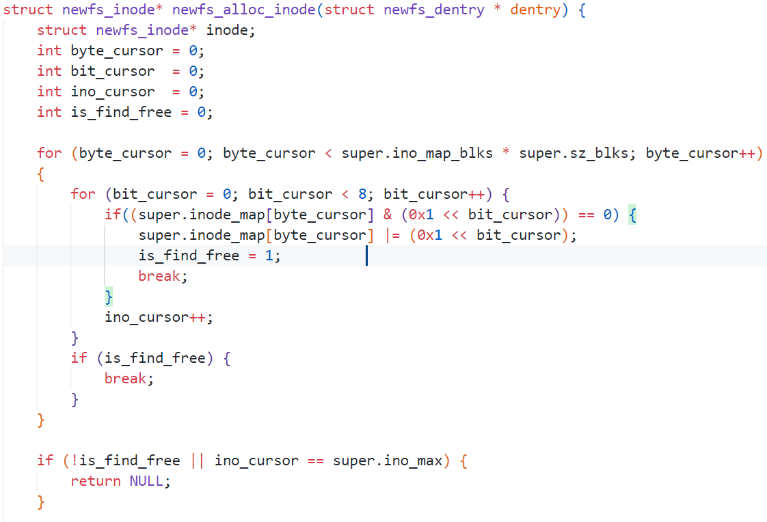
其中，new\_dentry 函数用于创建并初始化一个新的目录项。该函数接收目录项的名称和文件类型作为参数，为目录项分配内存并将其初始化为零， name、ftype、dno、ino、inode、parent 和 brother 字段被初始化：



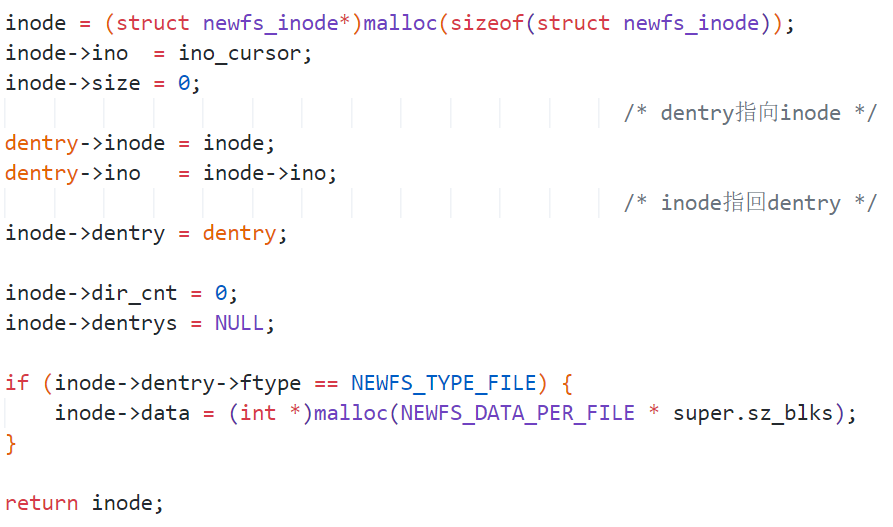
其中，newfs\_driver\_read函数用于从磁盘中指定偏移处读取指定大小的文件输出。该函数根据块大小对输入的偏移量进行向下和向上对齐，计算出需要读取的块数，再通过文件描述符读取相应的数据块，并将读取的内容拷贝到指定的输出缓冲区：



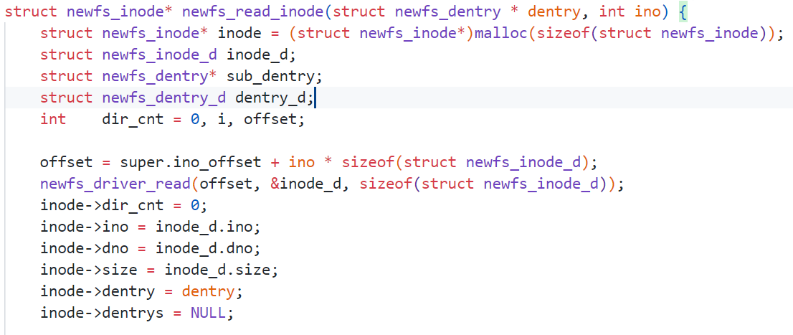
其中，newfs\_alloc\_inode函数用于分配并初始化一个新的 inode。该函数首先通过遍历 索引节点位图查找一个空闲的 inode，并设置其对应的位为已使用：



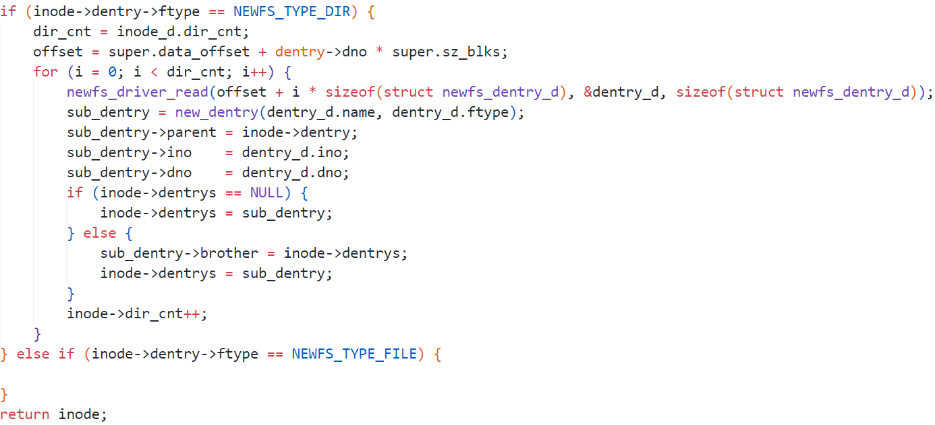
找到空闲inode后，函数为该inode分配内存，设置其基本属性，并将其与传入的 dentry 关联，dentry指向新分配的inode，而inode也指回对应的dentry。如果inode类型为文件，为该文件分配数据存储空间，最后返回指向inode的指针：



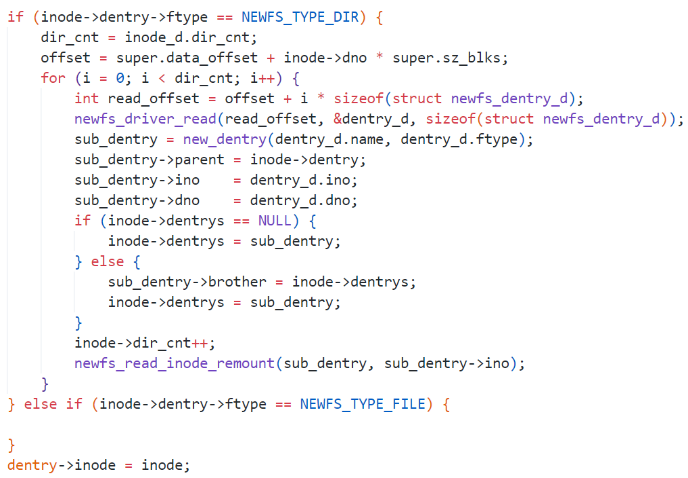
其中，newfs\_read\_inode函数用于从存储介质中读取指定inode的数据，并初始化一个新的inode结构体。函数首先从指定的 inode 位置读取其基本信息，并设置到新分配的 inode 结构体中：



如果该 inode 对应的是目录类型，则函数继续读取该目录下的所有子目录项，并为每个子目录项创建新的 dentry 结构体，将其链接到当前 inode 的 dentrys 列表中：



其中，newfs\_read\_inode\_remount函数用于未进行初始化时读取磁盘中的文件层次，基本结构与newfs\_read\_inode函数一致，只是增加了递归读取目录的过程调用：

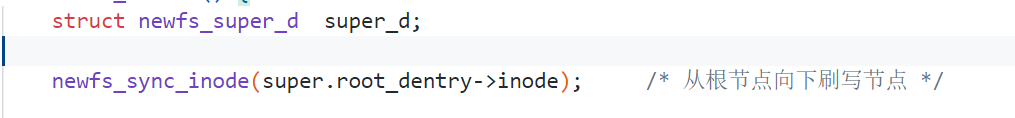


1. 卸载

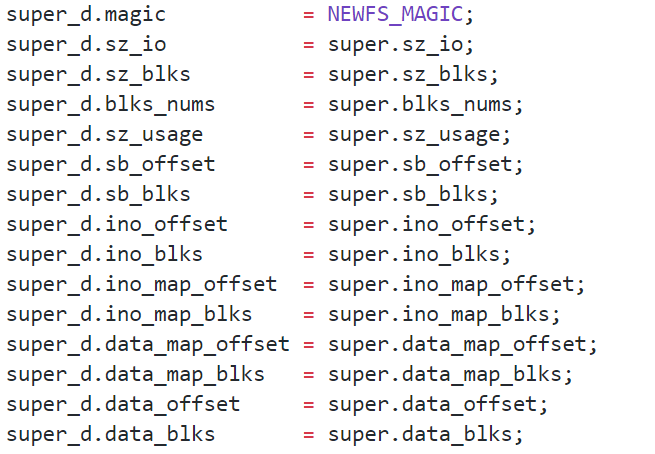


卸载过程调用辅助函数newfs\_umount：

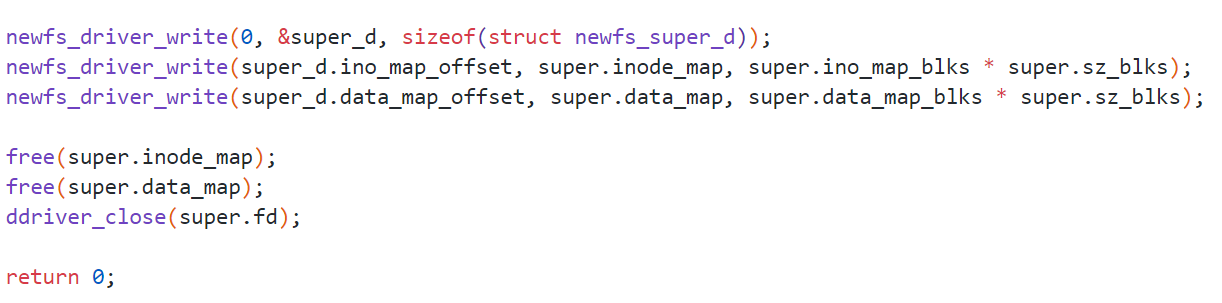
辅助函数中，首先从根节点开始向下将内存中的数据写回磁盘：



再填充写回超级块的信息：

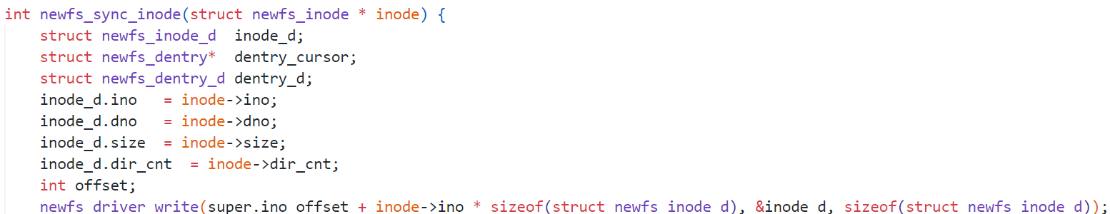


最后，写回超级块、索引位图和数据位图，释放位图变量的空间，关闭设备：

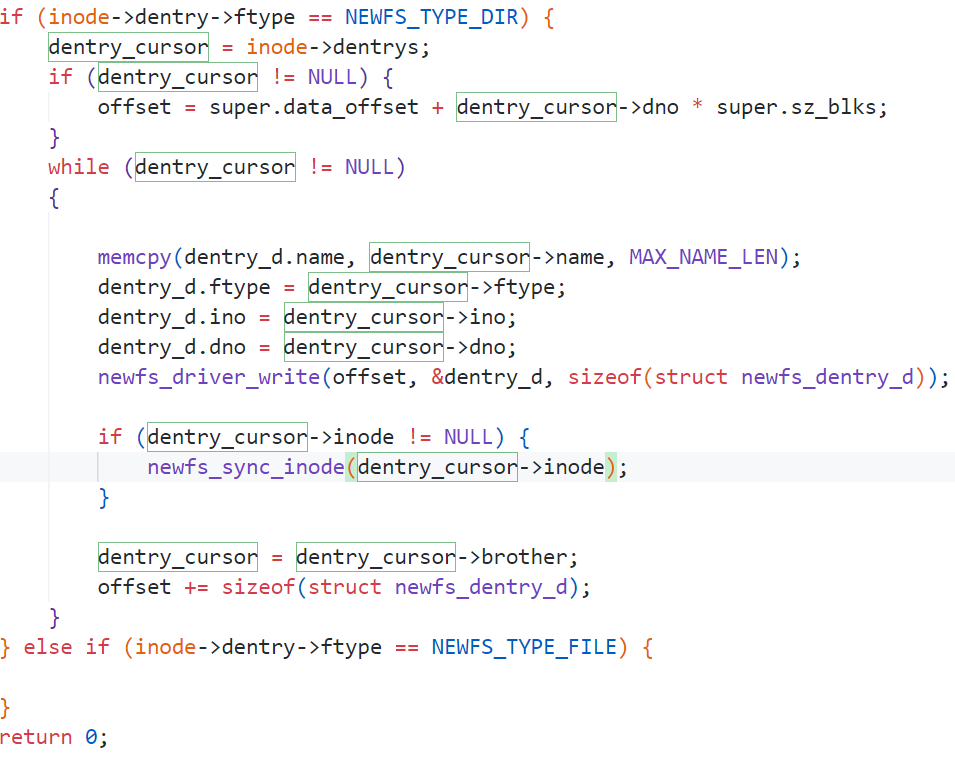


其中，newfs\_sync\_inode 函数用于同步 inode 及其相关的目录项数据到存储介质。

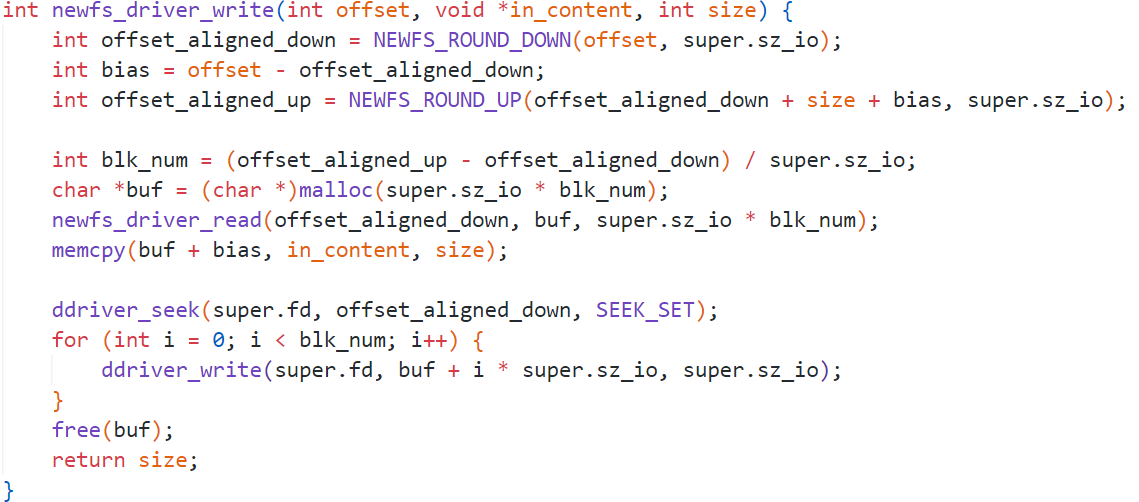
函数先将 inode 的基本信息写入磁盘：



如果该 inode 对应的是一个目录，函数会遍历目录中的所有目录项，并将每个目录项的详细信息写入磁盘。如果某个目录项对应的 inode 不为空，函数会递归调用自己来同步该 inode：



其中，newfs\_driver\_write函数用于将数据写入存储介质，并确保数据按照块对齐方式存储。该函数首先计算并对齐偏移量，然后读取目标位置的原数据，将传入的数据复制到缓冲区的相应位置，再将更新后的数据按块写回存储介质：

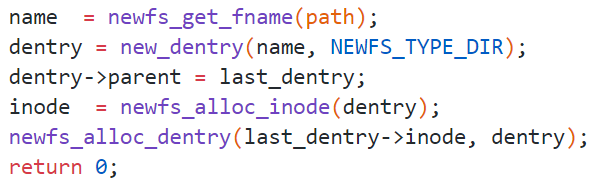


1. 创建目录

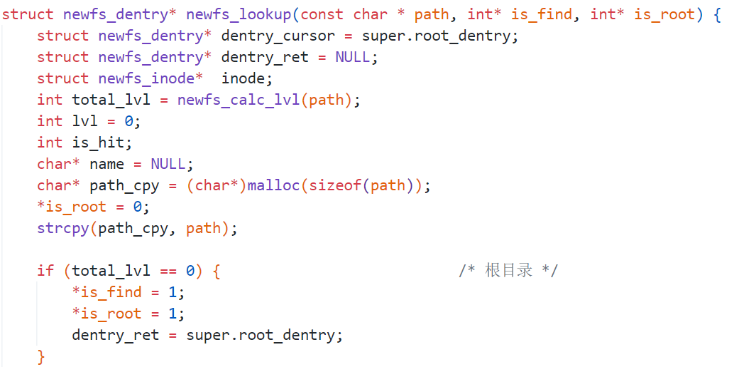
newfs\_mkdir函数使用newfs\_lookup查找路径中的最后一个目录项，如果该目录项已存在或者最后一个目录项对应的是一个文件而非目录，则返回错误码：



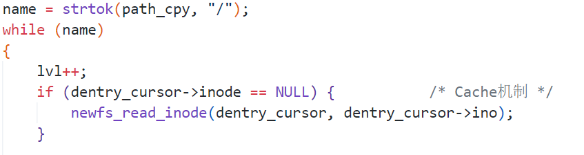
之后，函数从路径中提取目录名称，并创建一个新的目录项，调用 newfs\_alloc\_inode 为新目录分配一个inode，并通过 newfs\_alloc\_dentry 将该目录项添加到父目录的目录项列表中：



其中，newfs\_lookup函数函数用于根据路径查找文件系统中的目录项。函数首先计算路径的层级数，并复制路径字符串。如果路径层级为 0，表示查询的是根目录，直接返回根目录项并标记为找到：



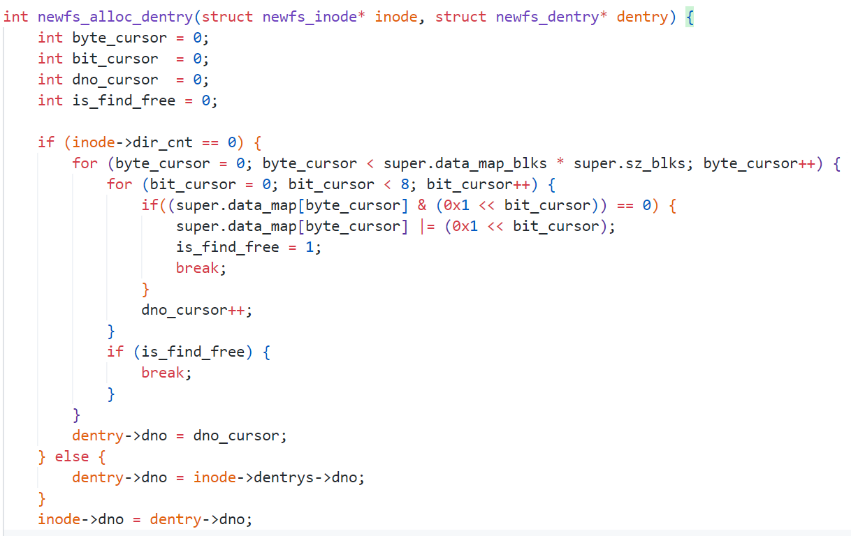
如果路径层级大于 0，该函数使用 strtok 分割路径，根据每个路径组件逐级查找对应的目录项。在查找过程中，若遇到缓存中未加载的 inode，会调用 newfs\_read\_inode 进行加载：



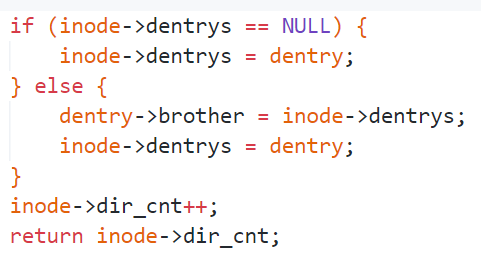
接下来，该函数会遍历每一级目录的子目录项，直到找到匹配的目录项。如果在某一级未找到匹配的子目录项，则标记为未找到并退出。若找到目标目录项，函数会返回该目录项并更新 is\_find 为 1。如果路径中的某个部分是文件而非目录，函数会提前退出并返回错误。最终，newfs\_lookup 返回找到的目录项（dentry\_ret），并且如果该目录项的 inode 尚未加载，函数会进行加载：



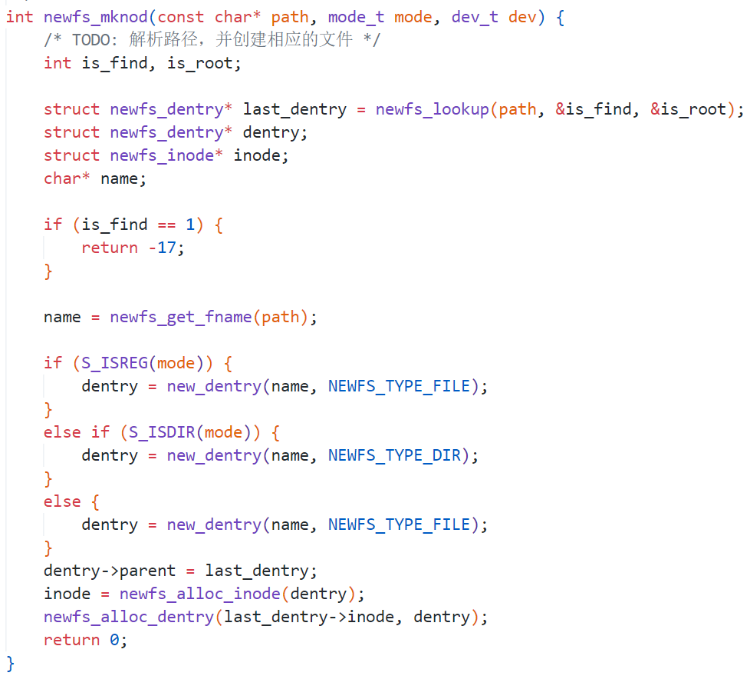
其中，newfs\_alloc\_dentry函数用于为给定的 inode 分配一个新的目录项。首先，函数检查 inode 的 dir\_cnt 是否为 0，如果是，则意味着该 inode 尚未分配任何数据块，则该函数遍历数据位图，查找第一个空闲的数据块，并将该块标记为已使用，分配给新目录项，如果该 inode 已有目录项，则直接为目录项分配第一个已有的数据位图标记：



接着，函数将目录项添加到 inode 的 dentrys 链表中，若该 inode 的 dentrys 为 NULL，则直接将新目录项作为首个子目录项。最后，更新 inode 的 dir\_cnt 并返回目录项数量：



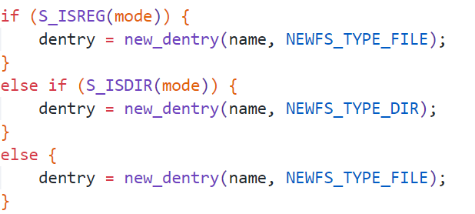
1. 创建文件



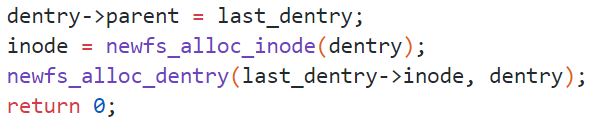
newfs\_mknod函数使用 newfs\_lookup 查找给定路径的最后一个目录项，并检查该路径是否已存在。如果路径已存在，返回错误码：



然后，函数提取路径中的文件名，并根据给定的模式（mode）判断要创建的是文件还是目录：

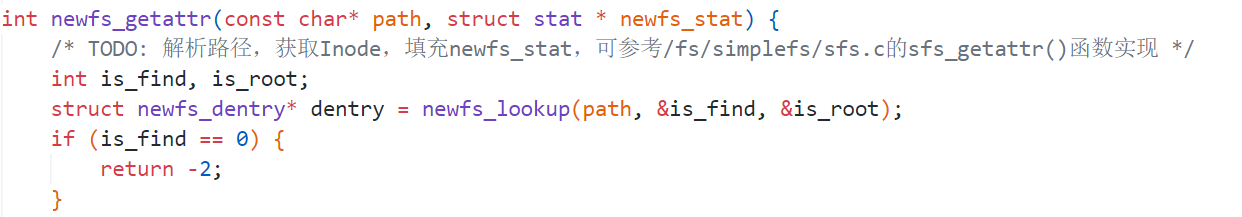


最后，将新创建的目录项与父目录关联，并为其分配一个 inode，再调用 newfs\_alloc\_dentry 将新目录项添加到父目录的目录项列表中：

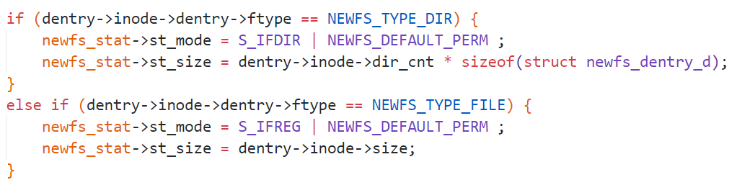


1. 显示文件

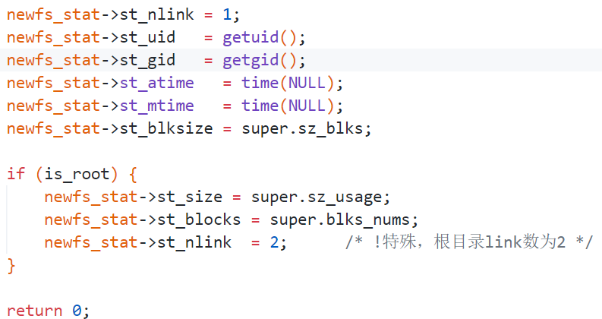
newfs\_getattr函数首先调用 newfs\_lookup 查找路径对应的目录项。如果路径未找到，返回错误码：



然后，根据目录项类型（文件或目录）来设置 newfs\_stat 的属性

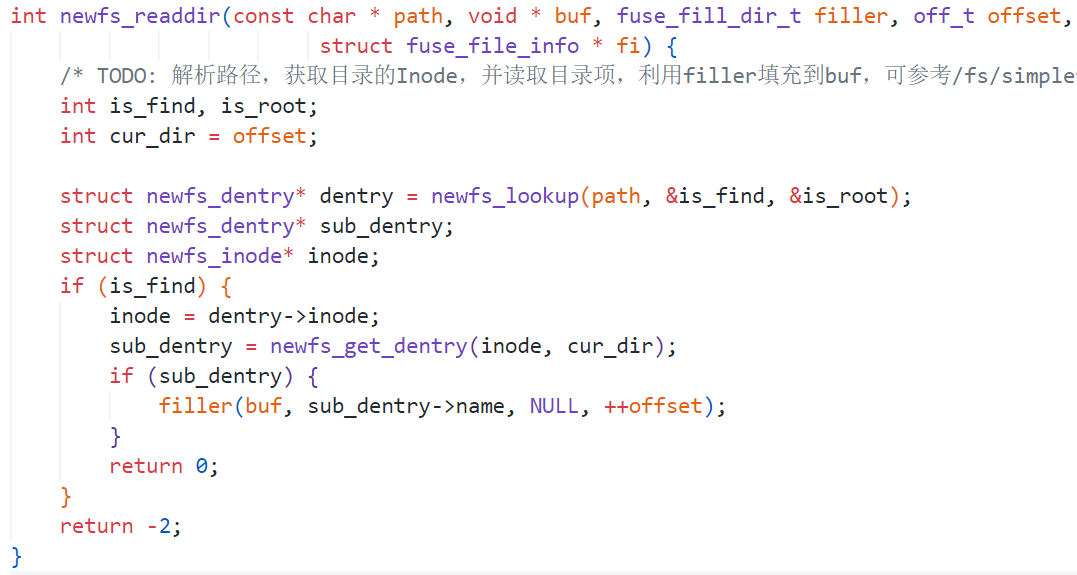


接着，设置其他属性，包括链接数、文件所有者的用户ID和组ID、访问时间和修改时间，以及块大小等：



1. 读取目录

newfs\_readdir函数首先使用 newfs\_lookup 查找给定路径对应的目录项，并获取该目录项的 inode。如果路径被找到，函数通过 newfs\_get\_dentry 获取指定偏移量下的子目录项，再使用 FUSE 提供的 filler 函数将子目录项的名称添加到 buf 中，填充目录的内容。为了支持目录遍历，offset 会在每次填充时递增：



1. **实验特色**

*实验中你认为自己实现的比较有特色的部分，包括设计思路、实现方法和预期效果。*

1. 实现了文件的灵活读写，即调用读写文件的函数前无需先移动磁盘头对齐和读写大小可以不固定。
2. 采用递归的方式在重新挂载时读取文件的层级结构。
3. 在申请时按需分配索引节点和数据块。

# 二、遇到的问题及解决方法

*列出实验过程中遇到的主要问题，包括技术难题、设计挑战等。对应每个问题，提供采取的****解决策略****，以及解决问题后的效果评估。*

1. 问题：不了解怎么对文件系统进行操作，怎么设计文件系统。

解决方法：仔细研究simplefs的代码，熟悉文件系统功能实现的主要操作的细节。

1. 问题：数据区写回位置错误。

解决方法：断点调试，逐个文件检查其写回位置和其在数据位图被分配的位置是否一致，最终发现是挂载时数据区偏移设置有误。

1. 问题：重新挂载后无法读取到文件。

解决方法：断点调试，检查重新挂载后文件层次结构建立的过程，发现递归建立结构时数据索引计算出错，更改递归逻辑和索引计算得以解决。

# 三、实验收获和建议

*实验中的收获、感受、问题、建议等。*

简易的ext2文件系统实验让我对操作系统中文件系统的核心原理和实现细节有了更深入的理解。通过实现inode结构、目录项、数据块分配等功能，我对磁盘上的文件是如何被组织、存取、修改等有了更清晰的认识。特别是在inode的设计上，我意识到它是文件系统中至关重要的组件，它不仅存储文件的元数据，还通过指针链接到数据块，管理文件的内容和位置。在实现目录项和目录结构时，我体会到了如何高效地组织文件和文件夹结构，以便快速地检索和定位文件。同时，位图的设计和实现让我更加理解了磁盘块的分配与回收，如何避免碎片化并最大化磁盘利用率。

目前整体实验的测试用例还是比较少，后续可以在测试中添加建立非空文件，增加更多的重新挂载测试用例。

# 四、参考资料

*实验过程中查找的信息和资料*

《simplefs文件系统参考实现》https://os-labs.pages.dev/lab5/part6/