openEuler base

openEuler是一款开源操作系统。当前openEuler内核源于Linux，支持鲲鹏及其它多种处理器，能够充分释放计算芯片的潜能，是由全球开源贡献者构建的高效、稳定、安全的开源操作系统，适用于数据库、大数据、云计算、人工智能等应用场景。同时，openEuler是一个面向全球的操作系统开源社区，通过社区合作，打造创新平台，构建支持多处理器架构、统一和开放的操作系统，推动软硬件应用生态繁荣发展。

本次发行版本为 openEuler 1.0 的 Base 版本，即基础版本，包含了运行最小系统的核心组件。

openEuler kernel

The openEuler kernel is the core of the openEuler OS, serving as the foundation of system performance and stability and a bridge between processors, devices, and services.

源码网址

1. 代码仓: https://gitee.com/openeuler
2. 软件包仓: https://gitee.com/src-openeuler
3. openEuler kernel:https://gitee.com/openeuler/kernel

文件系统

以**kernel-openEuler-1.0-LTS**为例，文件系统主要位置在**/kernel-openEuler-1.0-LTS/fs**

众所周知，文件系统是Unix系统最基本的资源。最初的Unix系统一般都只支持一种单一类型的文件系统，在这种情况下，文件系统的结构深入到整个系统内核中。而现在的系统大多都在系统内核和文件系统之间提供一个标准的接口，这样不同文件结构之间的数据可以十分方便地交换。Linux也在系统内核和文件系统之间提供了一种叫做VFS（virtual file system）的标准接口。

一般函数

通过使用VFS，文件系统的代码就分成了两部分：上层用于处理系统内核的各种表格和数据结构；而下层用来实现文件系统本身的函数，并通过VFS来调用。这些函数一般包括：

* 管理缓冲区(buffer. c)。
* 响应系统调用fcntl() 和ioctl()(fcntl.c and ioctl.c)。
* 将管道和文件输入/输出映射到索引节点和缓冲区(fifo.c, pipe.c)。
* 锁定和不锁定文件和记录(locks.c)。
* 映射名字到索引节点(namei.c, open.c)。
* 实现select( )函数(select.c)。
* 提供各种信息(stat.c)。
* 挂接和卸载文件系统(super.c)。
* 调用可执行代码和转存核心(exec.c)。
* 装入各种二进制格式(bin_fmt*.c)。

注：/kernel-openEuler-1.0-LTS/fs目录下无fifo.c文件，可能其功能在别处实现。

官方说明文档在**/kernel-openEuler-1.0-LTS/Documentation/filesystems**

实验内容

实验目的

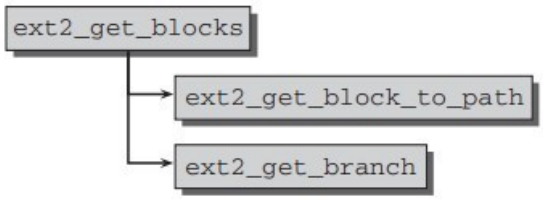
通过阅读源码了解**UNIX**文件系统**文件索引结构**,以**EXT2**第二代扩展文件系统为例。代码位置在**/kernel-openEuler-1.0-LTS/fs/ext2**，核心功能由**inode.c**实现，本次作业提示中内容并不完整，同学们同时需要在阅读中探索源码其他需要解读的部分。

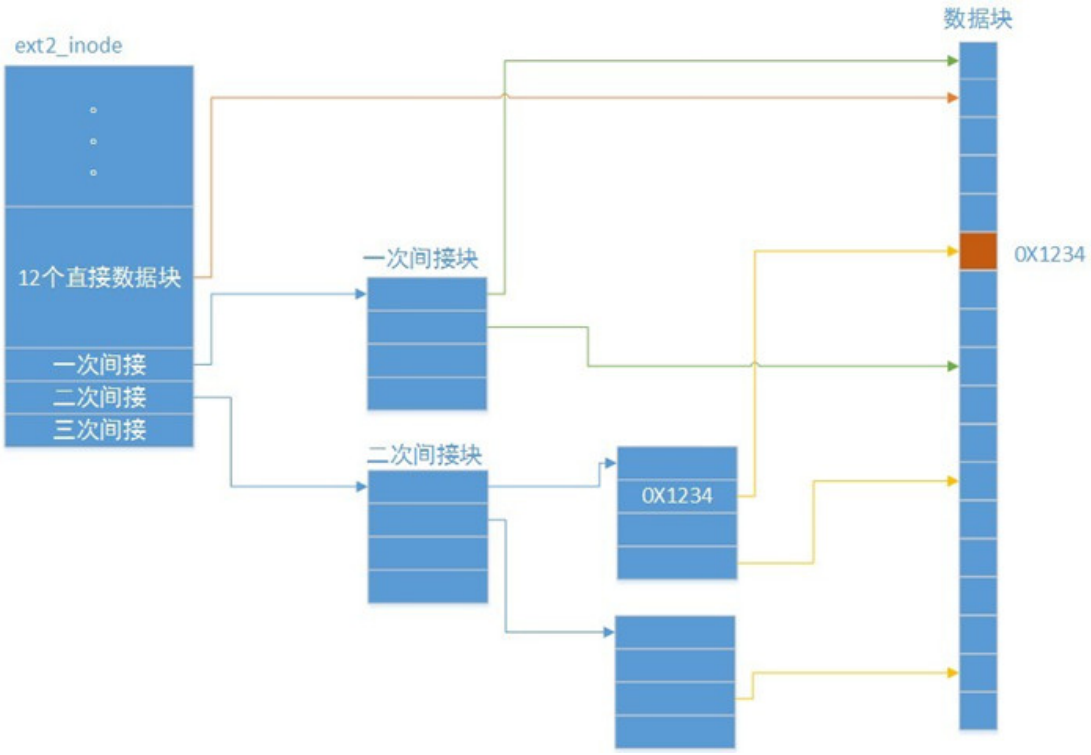
核心内容

- 要从硬盘上读文件，首先要找到文件数据所在的数据块， **ext2_get_block**实现的就是这个关键的功能。

  该函数是对函数**ext2_get_blocks**的封装。

- 要找到通向块的路径，就是确定该块是直接的还是间接的？是几次间接？每一次间接在间接块中的偏移量是多少？系统使用**ext2_get_block_to_path**获取块的定位信息。
- **ext2_get_branch**跟踪这个路径，最终到达一个数据块。





加强内容

需要向块设备写入数据时需要创建新块。创建新块需要两个信息：

1. 找到一个空闲块，作为新块。为了使同一个文件的数据块更为紧凑，内核会进行预留块操作。
2. 新块的地址指针存放在哪里？直接数据块？间接块？几次间接？

下面作简单提示：

- 搜索空闲的数据块了，由函数**ext2_find_goal**()完成。由于文件组织的紧凑性，请思考**如何确定目标块的位置。**
- **ext2_alloc_branch**函数调用**ext2_alloc_blocks**对给定的新路径分配所需的块。

- 实际上分配块的繁重的任务是由**ext2_new_blocks**完成的，**ext2_alloc_blocks**一直重复的调用它直到完成目的。

拓展内容

自行探索文件系统中的一般函数（前面有简单介绍）。