操作系统 lab4 实验报告

计算机系 2019011312 姚建竹

第六章

实现的功能

linkat

在Inode类中实现linknode函数，只有root会调用，给定源文件名称与目标文件名称。
需要将被链接的inode的id赋给新inode，然后获得被链接的inode的block id 和offset，生成一个新的inode。同时使用root_inode的increase_size，增加一个文件
在fs中检查如果oldname和newname不相同的话，调用ROOT_NODE的linknode函数，否则返回-1

unlinkat

在Inode类中实现unlink函数，只有root会调用，给定文件名称。循环遍历root内文件，找到名字后清空对应的DirEntry
实现get_link_num函数，root调用，给定block id offset，对比当前目录下有多少个文件有相同的block position，返回link的数量
当link数量为1时，unlink时需要调用clear()，否则只需要调用unlink即可

fstat

如果fd不合法，返回-1

根据fd_table[fd]进行类型转换（先转换成Any再转换成OSInode），得到对应inode的信息，从而获得link的数量，构建一个新的Stat类型，填好后写到对应的指针里即可。

问答题

在我们的easy-fs中，root inode起着什么作用？如果root inode中的内容损坏了，会发生什么？

ROOT INODE作为最顶层目录，有着至关重要的作用，可以进行创建文件，查找文件，获得文件列表，link文件，unlink文件，以及获得link文件数量的功能。总之，可以使用ROOT INODE进行各种easy-fs的操作。如果root inode中的内容损坏了，会影响我们对于文件的各

种操作。

第七章

问答题

举出使用 pipe 的一个实际应用的例子。

例子：ps –ef | grep name | grep –v grep | grep –v tmux | cut –c 5–16 | xargs kill –9

解释：通过使用管道，将每一个程序的输出作为另一个程序的输入，ps –ef得到对应进程的信息，grep获得含有name字段的进程信息，且排除掉grep和tmux，cut –c获得该行的5–16的字符，是进程id，然后作为args传入kill –9强行杀进程。

如果需要在多个进程间互相通信，则需要为每一对进程建立一个管道，非常繁琐，请设计一个更易用的多进程通信机制。

可以通过互传信号来解决
比如一个进程想要和其他进程发生通信，可以通过信号异步通知其他进程一个特定事件已经发生，需要及时处理。当一个信号发送给某个进程且定义了信号的处理函数，将会处理这个信号，响应完这个事件后还可以继续恢复前面的工作，从而简化了应用程序响应事件的开发工作，没有管道那么繁琐。而当没有信号时，进程可以做自己的事情，实现异步通信，比较高效。
而对于信号之外想要传递的数据，可以在内核里面设置一个各个进程都可以使用的管道缓存区，如果有进程想要使用，具体空间由内核进行分配，并通过信号类别与通信进程id一起来索引。