自定义 ls 和 wc 指令实现和源码对比

171250632 林哲远

1. 程序功能概述

ls -l (-d, -R, -a, -i) 显示当前目录下的文件及其相关信息（默认指令为-l），-d：仅显示当前目录，-i：显示 inode，-a：显示隐藏文件和隐藏文件夹，-R：递归显示当前目录下所有子目录文件夹

wc [filename] 统计文件的单词数、行数、所占字节数，多文件下，会分别输出文件统计数据和总和数据

2. 实现思路

1) ls

1. argv 用于提供参数（-d, -R, -a, -i）
2. 调用 file_operation() 方法进行处理，根据首先判断是否存在-d，如果存在就不继续进行，直接输出本身目录结构并终止
3. 如不是-d，进入 recursion()中，分别判断 Rai 的命令是否存在并根据情况进行输出。-R 先输出一个文件夹名，-a 输出包括./和../在内的隐藏文件夹和隐藏文件。-i 在每个文件前输出其所在 inode。对所有文件，都会读取 stat，输出其可读写权限，链接数所属用户和最后修改时间，针对不同的文件类型，支持颜色输出强调，并会输出文件夹下文件占用块的数量。
4. file_mode 读取 stat 结构体判断权限，并根据类型返回颜色编号。File_giu_uid 根据 passwd 和 group 获取文件用户和用户组指针并输出。
5. 输出结束后会递归搜索下属文件夹

2) wc

1. 用 c 语言文件操作 fopen 获取文件内容，用 fgetc 根据行分隔符数量计算

总行数，通过空格换行等，通过存在的[字符,空格/制表符/换行符/文件终止]组合，判断存在的单词数目。通过 stat 结构直接获取每个文件占用大小

3. 代码比较
    1. 总体

        总的来说，代码写的抽象层级较高，和源码比起来，没有写底层文件的“感觉“，速度上也相对拖沓一些。”

    2. Ls

        Ls 源码支持按多种手段进行文件夹排序，这一点是通过 pending_dirs，用类似链表的方法进行了实现，并通过 sort_options 和 sort__files，使用自定义排序算法和 qsort 对链表进行排序，非常精巧，在自己的代码中我原本也尝试了链表排序的手法，但无奈数据结构没学好，链表排序时常出错崩溃，为了安全起见还是注释掉了。

        颜色是我根据文件类型，在 file_mode 中输出并表驱动赋值，这点类似 ls 源码，但他使用的是结构体的手法实现，我是用的字符串数组，一些和 regular file 颜色一致的文件如 unknown 等我均没有进行判断而是直接输出，不利于未来进行迭代和可扩展性。

        Ls 源码在文件读写的大方向上与我的实现相一致，但他考虑的更周全更详细，比如我没有计算文件名长度而是直接 100 长的数组，少数情况下，如超长文件名或多文件夹嵌套中可能存在溢出崩溃问题，不够安全。对于文件和文件夹都抽出了不同的函数，利于维护和扩展，我写的相对随意。

        源码支持本地化语言，但我这个只能支持库自带的英文和我自己写进去

的中文。

3. Wc

   Wc 实现中印象最深的就是计算词数时出现的 goto，我的计算方式是根据网上 https://blog.csdn.net/z_erduo/article/details/103373997 的改良（复杂化）来的，通过计算增加后的[字符,空格/制表符/换行符/文件终止]组合，计算出单词数量，而源码，用的 ISSPACE 函数直接获取了空字符并计算。这个倒也不算出格，但那一串 goto 属实让我回忆起了儿时汇编语言的快乐。