自定义ls、wc程序与源代码对比文档

131250047
华苏珺

一.目的

自己编程，简单实现 “ls” 和 “wc” 命令，主要实现如下功能与参数:
ls–l(-d,-R,-a,-i) [path]
wc [path]
wc接收目标流(文件或者是标准输入)，并且根据参数配置对目标流中的字符、字节等相关数据进行统计，并且打印出来。
ls将指定目录文件下的文件(可能包含子文件目录以及文件),以及相关信息打印出来。

二.详细设计

a) ls

1. argv[1]接受第二个参数(l，d，R，a，i)，具体调用不同的函数。
2. 命令实现均先通过 stat 方法读取文件的信息。
3.ls –l区分文件和文件夹，通过读取&st 结构体指向的具体信息，输出权限、链接数、时间等信息。
4.ls –R 命令通过subFile变量保存当前路径下的目录路径，同时一边输出当前目录的文件。
char subPath[1024][1024]
最后递归调用本身，输出子目录下的文件。

```
for(;num>=0;num - -){
      display_Subfile(subPath[num]);
}
```

5. ls -i 同样依靠读取 stat 的 st_ino 变量实现

```
 printf("%9ld %s\t", st.st_ino, dirent->d_name);
```

b) wc

1. wc 命令经过测试发现，分词主要依靠空白符实现，即\t、\v、\n、\r、\f 和空格，而行数主要是\n的个数，字节数可以读取 st_size。
2. 通过 line_count 和 word_count 方法读取分别计算文件的行数和词数。
3. 词数计算方法：分别记录上一个字符和当前字符，当且仅当当前字符为空白符时num++。

三.源码比较

a)ls比较

ls 命令，源码考虑了输出颜色、文件类型(unknown, fifo, chardev, directory, blockdev等)、软链接和硬链接等多方面差异，实现较为详细复杂。

在ls的实现中，需要关注的主要分为两块:
1.对于文件相关信息的获得，并且根据标志位(flag来决定显示出哪些信息)
2.对于给定目录，如何遍历其下的子目录以及子文件，更一般的情况是进行递归遍历(在Linux中的文件系统都是树形的)。

对于第一种数据，我们通过上面的stat结构体，以及对应的getpwuid以及getgrpid可以获得所有的相关信息; 对于第二种，我们需要实现对于目录的递归，此时需要标准库中的dirent.h下的DIR结构体，通过readdir等函数来读取某目录下的子文件，来实现递归。
最后,在递归的过程当中，我们只需要将打印文件信息的功能封装为一个函数，就可以通过传入入函数指针，完成我们需要的定制性功能了。回调函数会对operation的标准位进行检查，从而保证打印出我们所需要的信息。 在我的实现中，这些信息都是取出后直接存放在变量中，然后打印出来。在源码中，其定义了fileinfo的结构体，并且通过枚举类定义了与这些文件信息相关的类型。源码是通过一系列节点组合成的链表将文件信息组织起来。

b)wc比较

源码中wc的实现通过一次遍历，多个变量保存，而我使用了两个函数遍历了两次。wc源码的计算词数也是通过计算空白符个数，计算行数也是通过计算使用\n个数。 wc中还比较需要关注的是对于word以及char的定义，对于word而言，我定义的是以空格作为切分的简单字母组合，而关键点在于char，在源码中，对于字符，或者说宽字符的类型都是通过宏或者typedef来定义的，这对于wc在不同系统平台上的实现,或者对于不同字符编码集的字符宽度(Unicode-16或者ASCII), 这些是我的简单实现中没有涉及到的。