Linux 程序设计实验报告

141250017
陈自强
2017/03/21

一、实验目的

通过编程了解 Linux 内核代码的实现机制，对 Linux 有进一步的了解

二、实验内容

自己编程实现“ls”和“wc”命令，要求实现如下参数与功能。
ls -l(-d, -R, -a, -i)
wc [filename]
完成实现后，查找原命令的源代码，对其进行阅读分析，与自己的版本作比较，看有何不同，并将不同之处写入文档。

三、实验实现

本次实验采用 C++作为编程语言。

ls 命令：

ls 命令核心即为对目录下的文件调用访问方法，核心是通过 stat 方法获取一个文件/目录节点的信息。
核心方法为

```
int ls_l(const char *path, int mode, int omitSecretFile, bool showSerialNum)
```

其参数为：

- Path：路径
- Mode：显示模式，SIMPLE_MODE or DETAIL_MODE,决定是否输出文件详细信息
- omitSecretFile:是否忽略以’.’开头的隐藏文件
- showSerailNum:是否显示文件序列号

其他方法仅需要调用这个方法就可以实现了

ls -R 因涉及递归，单独实现
其实现机制是

1. 调用 ls 命令打印当前目录的文件

2. 调用“`vector<FileOrDir>* getAllSubFilesOrDir(const char* path)`”获取该目录下所有目录和文件
3. 对每个目录，递归调用自己，对每个文件，打印其信息

wc 命令：

wc 命令主要分为三部分，实现较为简单

- 获取文件字节数：调用 stat，获取 st_size 即可
- 获取文件行数:逐个读取文件字符，遇到'\n'计数即可
- 获取文件单词数量：逐个读取文件字符，以纯英文开始和结束的记为一个单词

四、与 Linux 源码实现的比较

wc 命令：

wc 命令我和源码实现的区别主要集中在以下部分：

- 健壮性：源码考虑了多个文件的复杂情况，考虑了平台问题
- 效率：我文件字节数和文件行数是通过读取文件两次实现的，效率较低，而源码只读取了一次，效率较高。且源码读取方式是 safe_read，效率较高，而我使用的是 fget，效率较低
- 完整性：wc 源码支持多文件，支持多种参数，而我实现的是一个简化版，仅支持一个文件参数，且无错误处理

ls 命令

ls 命令我和源码实现的区别主要集中在以下部分：

- 对于 ls -R 命令中尚未读取的目录，ls 源码采用一个链表结构处理，如下所示：

```
/* Record of one pending directory waiting to be listed.  */

struct pending
  {
    char *name;
    /* If the directory is actually the file pointed to by a symbolic link we
       were told to list, 'realname' will contain the name of the symbolic
       link, otherwise zero.  */
    char *realname;
    bool command_line_arg;
    struct pending *next;
  };
```

而我则声明了一个类作为存储，并使用 Vector 存储这些节点类

```
class FileOrDir {
public:
    string name;
    string path;
    bool isDir;

    FileOrDir(string _name, string _path, bool _isDir) {
        this->name = _name;
        this->path = _path;
        this->isDir = _isDir;
    }


};
```

- 源码中对于 ls -R 还做了很多工作，例如为了防止目录死循环（如 ls -aR 时当前目录'.'也会被输出），定义了一个数据结构存储已经访问过的节点

```
/* When using -R, initialize a data structure we'll use to
   detect any directory cycles.  */
if (recursive)
  {
    active_dir_set = hash_initialize (INITIAL_TABLE_SIZE, NULL,
                                      dev_ino_hash,
                                      dev_ino_compare,
                                      dev_ino_free);
    if (active_dir_set == NULL)
      xalloc_die ();

    obstack_init (&dev_ino_obstack);
  }

localtz = tzalloc (getenv ("TZ"));

format_needs_stat = sort_type == sort_time || sort_type == sort_size
  || format == long_format
  || print_scontext
  || print_block_size;
format_needs_type = (! format_needs_stat
                     && (recursive
                         || print_with_color
                         || indicator_style != none
                         || directories_first));
```

- 除此之外，还有一些不同：如 ls 源码支持多参数是定义了 mode 的，而我的这是分别调用子方法去实现，又如源码有考虑颜色输出，文件类型，软硬连接，转移字符等，实现较为复杂，不再赘述。