

# Linux 项目作业实验报告

蔡日骏 12348003

2014 年 12 月 24 日

## 1 实验简介

本次实验要求实现`ls`命令的基本功能，能通过`-l`参数切换简要/详细模式。

## 2 实现细节

### 2.1 遍历目录

Linux 下常用的遍历目录的方法有以下三种：

1. `opendir/readdir/closedir` 函数，使用比较麻烦，除非一边读取一边处理数据，否则需要动态分配内存并根据需要动态调整数组大小。
2. `scandir` 函数，使用方便，函数库会自动分配足够的内存，并能根据需要自动进行过滤和排序。
3. `fts_` 系列函数，功能强大，能方便地实现递归遍历，但不是 POSIX 标准函数。

最后我选用的是`scandir`函数，使用 `libc` 内置的`alphasort`函数作为比较器进行排序，并自定义一个过滤器函数用于过滤隐藏文件。调用方式如下：

```
struct dirent **namelist;
n_items = scandir(path, &namelist, _ignore_hidden, alphasort);
if(n_items == (size_t)(-1)) {
    perror(path);
    return EXIT_FAILURE;
}
```

过滤器`_ignore_hidden`用于过滤隐藏文件，定义如下：

```
int _ignore_hidden(const struct dirent *item)
{
    return !(item->d_name[0] == '.');
}
```

`scandir`会自动在堆上分配足够的内存，因此数据使用完后需要释放`namelist`数组中存放的地址以及`namelist`数组本身。

## 2.2 分栏对齐

ls 输出分栏时需要进行对齐。为保持整齐，对齐时计算每列最宽的行的宽度，该列就按照该宽度进行对齐。

在计算列宽时，一个需要处理的问题是，一个字符的字节长度并不一定等于其输出时的列宽。比如在 UTF-8 编码下，一个中文字符的字节长度为 3，但等宽字体下一个中文字符占 2 个英文字母的宽度。

为了解决这个问题，需要使用 `mbrtowc` 函数从多字节字符串中提取一个宽字符，然后使用 `iswprint` 和 `wcwidth` 函数计算该宽字符的输出宽度。为方便起见，定义了下面这个函数：

```
size_t mbs_pwidth(const char *s, size_t n)
{
    mbstate_t mbs;
    wchar_t wc;
    size_t clen, width = 0;
    const char * const s_end = s + n;

    memset(&mbs, 0, sizeof(mbs));
    while((clen = mbrtowc(&wc, s, s_end - s, &mbs))) {
        switch(clen) {
            case (size_t)(-1): /* truncated string (incomplete character) */
                s = s_end;
                ++width;
                break;
            case (size_t)(-2): /* invalid character */
                ++s;
                ++width;
                break;
            default: /* valid wide character */
                s += clen;
                if(iswprint(wc))
                    width += wcwidth(wc);
        }
    }

    return width;
}
```

为了使 C 的宽字符库正常工作，程序启动时需要使用 `setlocale(LC_ALL, "")`；进行本地化设置。

此外，在摘要模式中进行分栏输出时，为了计算需要分的栏数，还需要获取用户当前终端窗口的宽度。这个宽度可以通过 `ioctl` 系统调用进行获取。在获取失败时，使用默认宽度 80。定义下面这个函数：

```
int cal_max_width()
{
    struct winsize ws;
    if(!ioctl(STDOUT_FILENO, TIOCGWINSZ, &ws) && ws.ws_col > 0)
```

```

    return ws.ws_col;
    return 80; /* fallback (fail to get the width of tty) */
}

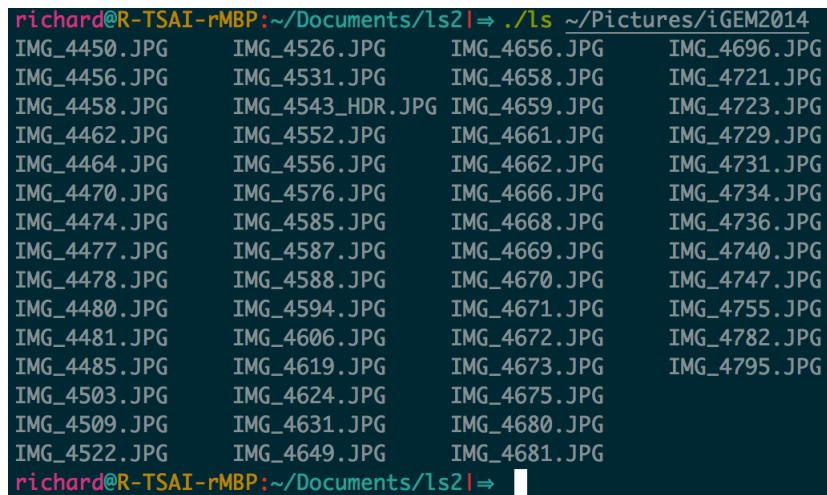
```

## 2.3 详细信息

在详细模式下 (提供 `-i` 参数), 使用 `lstat` 系统调用查询文件的详细信息, 并使用 `getpwuid` 和 `getgrgid` 函数获取 UID 和 GID 对应的用户名和组名。

## 3 演示

下面为程序运行效果。测试平台分别为 Mac OS X Yosemite (Darwin 14.0.0, clang 6.0) 和 Arch Linux (Linux Kernel 3.17.6, gcc 4.9.2)。

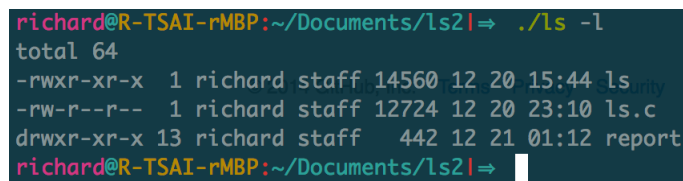


```

richard@R-TSAI-rMBP:~/Documents/ls2|⇒ ./ls ~/Pictures/iGEM2014
IMG_4450.JPG    IMG_4526.JPG    IMG_4656.JPG    IMG_4696.JPG
IMG_4456.JPG    IMG_4531.JPG    IMG_4658.JPG    IMG_4721.JPG
IMG_4458.JPG    IMG_4543_HDR.JPG IMG_4659.JPG    IMG_4723.JPG
IMG_4462.JPG    IMG_4552.JPG    IMG_4661.JPG    IMG_4729.JPG
IMG_4464.JPG    IMG_4556.JPG    IMG_4662.JPG    IMG_4731.JPG
IMG_4470.JPG    IMG_4576.JPG    IMG_4666.JPG    IMG_4734.JPG
IMG_4474.JPG    IMG_4585.JPG    IMG_4668.JPG    IMG_4736.JPG
IMG_4477.JPG    IMG_4587.JPG    IMG_4669.JPG    IMG_4740.JPG
IMG_4478.JPG    IMG_4588.JPG    IMG_4670.JPG    IMG_4747.JPG
IMG_4480.JPG    IMG_4594.JPG    IMG_4671.JPG    IMG_4755.JPG
IMG_4481.JPG    IMG_4606.JPG    IMG_4672.JPG    IMG_4782.JPG
IMG_4485.JPG    IMG_4619.JPG    IMG_4673.JPG    IMG_4795.JPG
IMG_4503.JPG    IMG_4624.JPG    IMG_4675.JPG
IMG_4509.JPG    IMG_4631.JPG    IMG_4680.JPG
IMG_4522.JPG    IMG_4649.JPG    IMG_4681.JPG
richard@R-TSAI-rMBP:~/Documents/ls2|⇒

```

图 1: Mac OS X Yosemite 1



```

richard@R-TSAI-rMBP:~/Documents/ls2|⇒ ./ls -l
total 64
-rwxr-xr-x  1 richard staff 14560 12 20 15:44 ls_ruby
-rw-r--r--  1 richard staff 12724 12 20 23:10 ls.c
drwxr-xr-x 13 richard staff   442 12 21 01:12 report
richard@R-TSAI-rMBP:~/Documents/ls2|⇒

```

图 2: Mac OS X Yosemite 2

```

→ /tmp ./ls *
ls  ls.c

yaourt-tmp-richard:
aur-libtinfo PKGDEST.nkq

ycm_build.0utLYE:
BoostParts          CMakeCache.txt          CMakeFiles          cm

ycm_temp:
server_58214_stderr.log server_58214_stdout.log server_59353_stderr.
→ /tmp █

```

图 3: Arch Linux 1

```

→ /tmp ./ls -l *
total 64
-rwxr-xr-x 1 richard users 15476 12 21 00:50 ls
-rw-r--r-- 1 richard users 12724 12 21 00:55 ls.c

yaourt-tmp-richard:
total 0
drwxr-xr-x 2 richard users 80 12 20 16:29 aur-libtinfo
drwx----- 2 richard users 40 12 20 16:29 PKGDEST.nkq

ycm_build.0utLYE:
total 280
drwxr-xr-x 3 richard users 100 12 20 16:41 BoostParts
-rw-r--r-- 1 richard users 15846 12 20 16:41 CMakeCache.txt
drwxr-xr-x 5 richard users 280 12 20 16:41 CMakeFiles
-rw-r--r-- 1 richard users 1622 12 20 16:41 cmake_install.cmake
-rw-r--r-- 1 richard users 107045 12 20 16:41 compile_commands.json
-rw-r--r-- 1 richard users 8328 12 20 16:41 Makefile
drwxr-xr-x 4 richard users 120 12 20 16:41 ycm

ycm_temp:
total 16
-rw-r--r-- 1 richard users 917 12 20 15:19 server_58214_stderr.log
-rw-r--r-- 1 richard users 0 12 20 15:19 server_58214_stdout.log
-rw-r--r-- 1 richard users 917 12 20 15:08 server_59353_stderr.log
-rw-r--r-- 1 richard users 0 12 20 15:08 server_59353_stdout.log
→ /tmp █

```

图 4: Arch Linux 2