选课序号：50

实验二：用C语言实现简单的Shell

姓名：杨凡

学号：2220172330

1. 实验目的：

理解shell程序的功能

学会Shell的使用

用C/C++构造一个简单的Shell

1. 实验内容：

基本任务1：用C/C++编写一个简单的Shell程序，实现以下基本的命令

浏览目录和文件的各种属性：ls

回显命令：echo

显示文件内容cat

创建目录：mkdir

删除文件或目录：rm

切换目录：cd

显示当前目录：pwd

文字统计：wc

基本任务2：用makefile文件管理该Shell程序

基本任务3：写清楚编译和运行的过程

选做任务：

增加help命令，简单查看该Shell程序提供的命令选项

增加history命令，查看之前使用过的前10个命令

1. 实验步骤：

实验总体框架：

MainFramwork.c中包含主函数，实现的是Shell程序的实现大体流程

每一个命令对应一个C文件，每一个C文件中是一个或者几个具体实现该命令的代码，有shell\_cat.c，shell\_cd.c，shell\_echo.c，shell\_help.c，shell\_ls.c，shell\_mkdir.c，shell\_pwd.c，shell\_rm.c，shell\_wc.c

Head.h头文件中是对MainFramwork中调用的每一个命令具体实现的函数的声明

Makefile文件：先对除了MainFramwork.c这个包含主函数的文件之外的C文文件进行编译生成汇编文件，然后对所有汇编.o文件连同MainFramwork.c文件进行链接生成一个可执行文件。然后再通过自定义的makefile命令clean删除所的汇编.o文件。

实验中的具体实现：

**MainFramwork.c**文件，在这个文件中有三个函数，mian()函数是程序入口，负责循环程序并每次显示命令输入提示符，并循环调用analysis函数。Analysis()函数负责将每次循环输入的字符串按照空格来分割，并保存在argc和argv[]中，在函数末尾调用do\_exec函数，并把argc和argv[]作为每一个命令函数的传入参数。最后是do\_exec（）函数，该函数根据传入的字符串数组的第一个字符串来判断应该调用哪个命令执行函数，并继续将两个参数传入命令执行函数。

**Head.h**头文件，在这个文件中是对在MainFramwork.c文件中直接调用的命令执行函数的声明shell\_cat（），shell\_cd（），shell\_echo（），shell\_help（），shell\_ls（），shell\_mkdir（），shell\_pwd（），shell\_rm（），shell\_wc（），并且用预处理指令：#ifndef，#define，#endif来确保函数声明不会被重复编译。

**Shell\_ls.c**文件，该文件用来实现linux下的ls命令，包含他的两个参数-a和-l，对于ls命令，首先需要了解DIR结构体，dirent结构体，和stat结构体。

函数DIR \*open(const char \*)打开一个文件夹，然后调用函数dirent \*readdir(DIR \*)从文件夹中依次读取其中的文件，每读取一个文件就会返回dirent结构体，里面包含着该文件的基本信息（包括d\_ino索引节点号，d\_off在目录文件中的偏移，d\_reclen文件名长，d\_type文件类型，文件名name）,但是对于ls -l命令而言，还需要用到stat结构体来-获得文件更加详细的命令，通过调用stat(char \*，stat)函数，获得指定文件的stat结构体，该结构体中包含更多的文件信息，包括st\_mode：文件对应的模式：文件，目录等，st\_ino：inode节点号，st\_dev：设备号码，st\_rdev：特殊设备号码，st\_nlink：文件的连接数，st\_uid：文件的所有者，st\_gid：文件所有者对应的组，st\_size：文件对应的字节数，st\_atime文件最后被访问的时间，st\_mtime：文件内容最后被修改的时间，st\_ctime文件状态改变时间，st\_blksize文件内容对应的块大小，st\_blocks文件内容对应的块数量。通过这些内容就可以基本写出ls -l命令。

**Shell\_cat.c**文件，该文件用来实现linux下的cat命令，包括-n参数，对于cat命令，采用最简单的putc和getc相结合的手段，从文件开头一个字符一个字符地读取，然后存放于stdout中，在换行和文件结束时打印出来。对于cat -n命令，就需要一行一行地读取，在打印的时候在每一行前加上行号。

**Shell\_cd.c**文件，该文件用来实现linux下的cd命令，其后不仅要能接上文件路径，还需要能接上-（表示上一次到达的目录）和~（起始家目录）。对于cd命令，需要POSIX标准操作系统API头文件unistd.h下的函数chdir(char \*)，对于一般路径和cd ~而言，直接指定路径即可，而对于cd -而言，需要用两个全局变量保存上一个路径和当前路径，在cd操作末尾刷新这两个变量。

**Shell\_echo.c**文件，该文件用来实现linux下的echo命令，对于echo命令，就是要打印echo后的内容，不论内容是否含有空格，都一并打印出。所以，用while循环，根据argc的大小来打印。

**Shell\_mkdir.c**文件，该文件用来执行linux下的mkdir命令，该命令创建文件夹，直接调用POSIX的API函数mkdir(char \*，[权限])，并将权限参数设为S\_IRWXU.

**Shell\_pwd.c**文件，该文件用来执行linux下的pwd命令，pwd命令不接参数，用来显示当前的路径。直接调用API函数getcwd(char \*)函数返回当前路径并打印。

**Shell\_rm.c**文件，该文件用来实现linux下的rm命令包括他的一个参数-r，rm命令的作用是删除文件文件或者文件夹，C语言中有一个unistd头文件下的API函数remove函数可以删除普通文件以及空文件夹，但是对于本身包含其他文件或者文件夹的文件夹却不能够直接删除，所以，对于这类文件夹，我们需要用到递归。还有一个值得注意的是，rm命令只能删除文件，而rm -r命令可以删除文件和文件夹，不论该文件夹里面包不包含文件或者文件夹。我们在文件中声明一个独立的函数remove\_dir（char \*path）用来删除目录路径是path下的文件或者文件夹。在该函数中秒首先需要判断时文件还是文件夹，如果是文件就直接滴哦安永romove函数删除，并作为该递归的出口。如果是文件夹，就调用remove\_dir（char \*path）函数进行递归处理，不过此时传入的参数path需要进行递归处理。

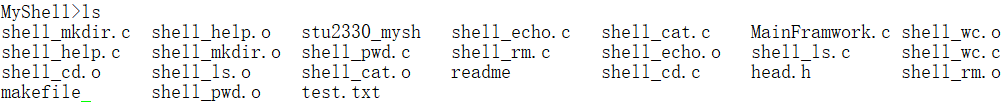
**Shell\_wc.c**文件，该文件用来执行linux下的wc命令，以及三个参数-l，-c，-w，wc命令的这三个参数，分别是显示对应文件的行数，字符数，单词数。如果wc命令不带参数，那么默认将会是等同于wc -lcw的形式，及三个信息都会显示。 我们在文件中额外定义三个函数wcline(char \*)，wcword(char \*)和wcchar(char \*)，分别是用来实现文件行数，文件单词数和字符数。然后在shell\_wc文件主函数中根据传入的参数按照需要来调用三个函数来是实现统计文件的功能。其中大体的统计思路是，统计函数，从文件头依次读取文件内容，如果遇到“\n“符，就将计数+1。对于统计字符数，也是从文件头开始一次读取字符，然后读取一个计数+1，知道文件结束符EOF。对于统计单词数，同样是从文件开头开始读取字符，只要是不包含空格的连续字符串就算做是一个单词，计数+1.

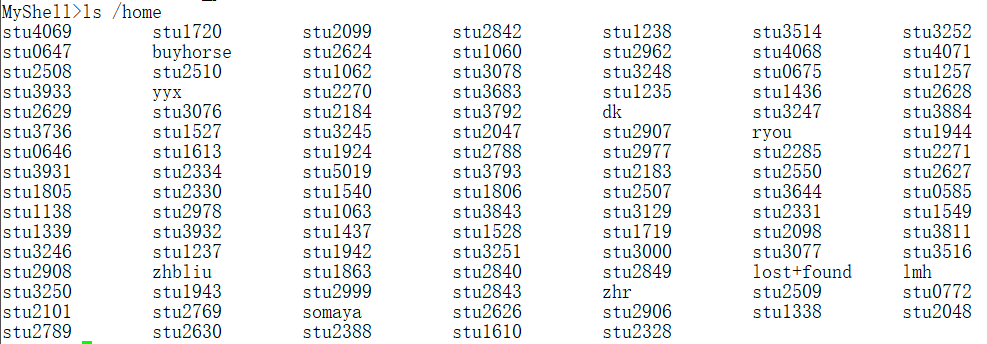
**Shell\_help.c**文件，该文件用来实现linux下的简化帮助命令，根据传入的参数会显示简单的命令帮助信息，包括命令功能，命令格式，命令参数及对应功能。

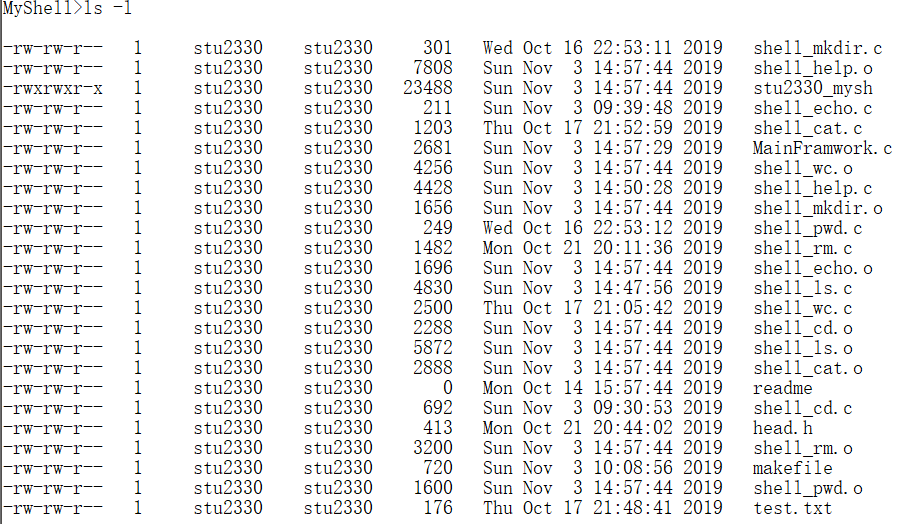
**Makefile** 文件：该文件用来按照预处理，汇编，编译，链接的步骤来将C源文件，C头文件生成一个头文件，同时管理整个Shell程序。先将除了MainFramwork.c之外的C源文件直接一步生成.o目标文件（当然中间经过了预处理、汇编和编译的过程），然后将所有目标文件连同MainFramwork.c文件一起链接成一个可执行的文件。

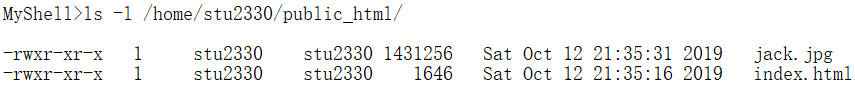
1. 实验结果：

**ls 命令：**

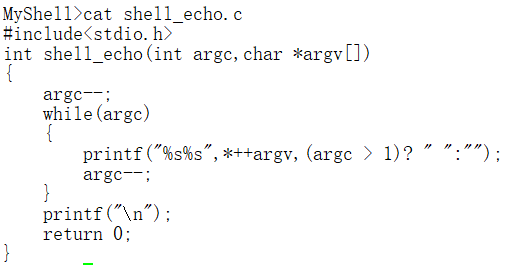




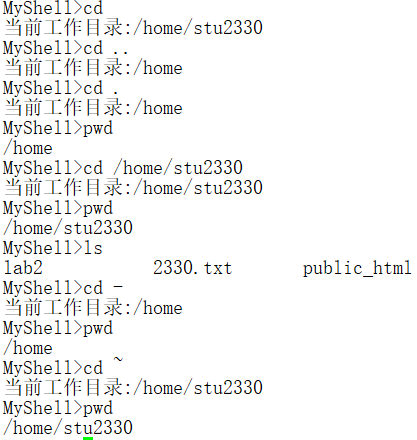




**cat 命令：**



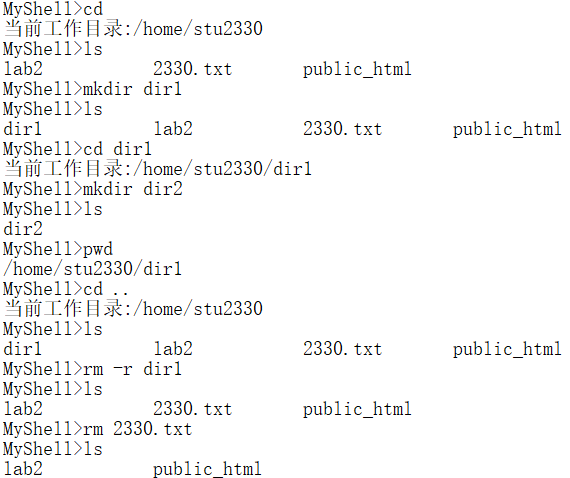
**cd 命令和pwd命令：**



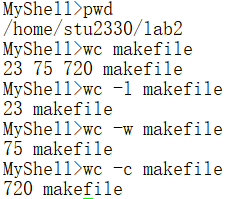
**echo 命令：**



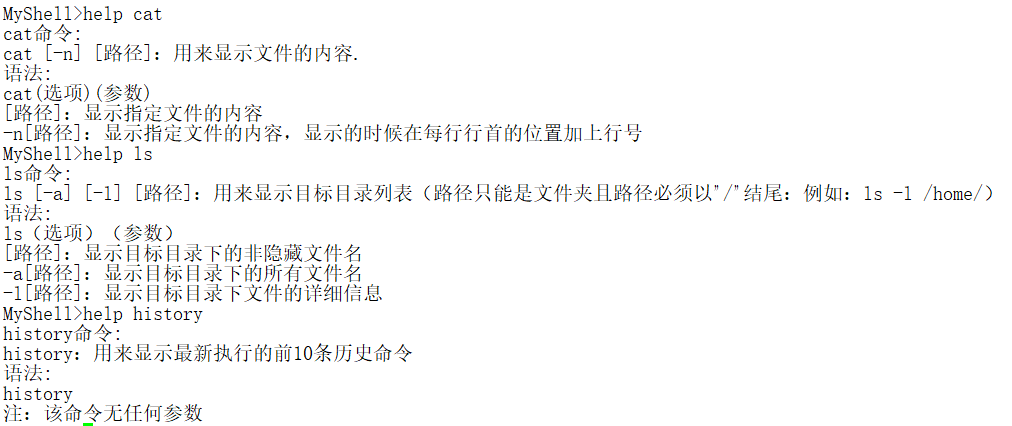
mkdir命令和rm命令：



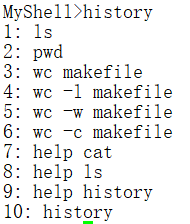
**wc 命令：**



**help 命令：**



**history 命令：**



1. 实验分析：

根据实验结果可知，实验包括的命令的基础功能大部分都能实现，但是在实验后期发现更高阶的功能往往是依赖于一些基础命令的功能的，这决导致如果我在完成前面简单的基础命令得到时候就应该要有意识的去考虑一些后续工作。举个例子：在考虑进阶功能（管道和重定向）的时候，我就发现这两个命令是将掐你给命令的输出作为输入，所以需要对前面所有含有输出的命令的输出结果做一个整合。

我的myshell程序对于命令行参数的输入顺序含有很大的限制，比如只能每次输入单个功能参数，且每个参数的位置都是固定的，如果要做到更灵活，还需要更改一些判断条件，由于时间问题，这一点我也没有做好。

在实际的linux系统中，每个命令的执行实际上是伴随着一个进程的执行，但在我的程序中，至始至终只有一个进程，虽然表面上也能实现shell命令，但如果深入linux底层去看，就会出问题，比如后台运行“&”程序，那么就必须要用到进程的概念，这一点也是我在设计程序时的一个失误。

1. 程序所有功能及使用方法：

注意：所有命令如果有功能选项参数，必须位于argv[1]，且不可同时带两个功能选项参数：如ls -a -l [路径]或者ls -al [路径]

ls [-a] [-l] [路径]：用来显示目标目录列表（路径只能是文件夹且路径必须以”/”结尾：例如：ls -l /home/）

[路径]：显示目标目录下的非隐藏文件名

-a[路径]：显示目标目录下的所有文件名

-l[路径]：显示目标目录下文件的详细信息

cat [-n] [路径]：用来显示文件的内容

[路径]：显示指定文件的内容

-n[路径]：显示指定文件的内容，显示的时候在每行行首的位置加上行号

cd [路径]|~|-：用来切换工作目录

[路径]：切换到指定路径

-：切换到上一个工作目录

~：切换到家目录

echo [字符串(可带空格)]：用来直接输出指定的字符串

[字符串]：直接打印出指定的字符串

mkdir [路径]：用来创建文件夹

[路径]：创建指定路径的文件夹

pwd：用来显示当前工作路径

注：该命令无任何参数

rm -r [路径]：用来删除一个文件或者一个文件夹

[路径]：删除指定路径的文件

-r [路径]：删除指定路径的文件夹

wc -c -l -w [路径]：用来统计一个文件的行数，字符数和单词数

-c [路径]：统计一个文件的字符数

-l [路径]：统计一个文件的行数

-w [路径]：统计一个文件的单词数

help [命令名]：用来提命令帮助

[命令名]：显示指定命令名的基本帮助信息

history：用来显示最新执行的前10条历史命令

注：该命令无任何参数

1. 编译方法

在/home/stu2330/lab2/目录下，执行make命令，即可完成编译，并生成“stu2330\_mysh”可执行文件。

运行方法

在/home/stu2330/lab2/目录下，执行./stu2330\_mysh命令，即可开始运行可执行文件

1. 设计文档

目的：

设计Shell程序的体系结构，模块及模块间的关系。从多种实现该目的的不同方案中选择最佳的。

运行环境：

Ubuntu Linux运行环境

需求概述：

1. 实现Shell的几个基本命令
2. 尽可能地尽可能还原Linux的原始命令功能
3. 提供友好的人机界面：例如提供命令帮助，历史命令列表
4. 用makefile文件完成管理项目的工作

系统分析：

这个模拟Shell的程序用来尽可能地还原Linux的自带bash的功能。基础思想就是循环+选择，但对于子选择的执行，有两种不同的方案：

1. 单独编写每个命令的.c文件，然后单独地将每个.c文件编译生成一个可执行文件，然后在MainFramwork.c中，在每个if中根据传入的命令行参数用exec族函数调用每个命令的可执行文件。
2. 同样单独编写每个命令的.c文件，但是不将其单独编译为可执行文件，而是在MainFramwork.c文件中根据if条件去调用每个单独命令C文件中的函数，以达到执行命令的目的。

这两种方法理论上都可以实现，但在makefile的管理上第一种方法不是很方便，并且如果采用单独的进程的话，那么多进程之间的信息交流也是一个比较麻烦的事。综上所述，我们采用第二种方法。

模块设计：

该程序将每个命令单独分为一个功能模块，每个命令的功能模块中可能不止一个函数为其服务，所以，对于比较复杂的命令（例如含有多个功能参数的命令），那么我们需要将一个命令的模块再分为多个子模块。

用主模块来调用各个功能子模块，就能在循环中有选择地实现各个模块功能。

流程图：

1. 用户手册

本模拟Shell的程序支持的命令名：cat，cd，echo，help，ls，mkdir，pwd，rm，wc，history

执行命令“./stu2330\_mysh”，就能开始运行该程序，程序会循环显示 “ Myshell> “，只需按照“help [命令名称]”的格式来输入命令。

1. 注意手册

注意在运行ls -l的命令时，如果要指定地址的话，末尾必须带“/”，例如：ls -l /home/，因为要在程序中拼接地址。

注意该模拟shell程序只是一个简易shell外壳，仅包含极小部分shell命令，必须参照帮助命令来运行程序，程序可能会出现一些bug，希望您发现后及时反馈给我们，联系邮箱：anordinaryman@qq.com