# lab09 Mini Shell

软件工程 2018 级  1813075  刘茵

## 实验目的

- Implement a mini shell which is simlar to bash shell

## 实验分析

1. 为了实现 shell 的路径功能，需要保存一个全局变量表示路径
2. 为了从输入读取命令并执行，对每个命令需要 fork 一个子进程

## 实验结构

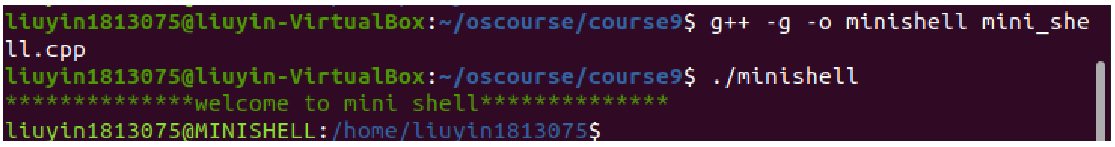
3. 主函数：输出欢迎语句，获取本地 shell 初始地址，获取输入，进行命令识别。
4. split 函数：讲读取的多个命令进行分割储存，返回 vector
5. eval 函数：运行命令，通过对关键词进行识别，区分 shell 内部和 bash 命令，分别进行处理和输出。

## 实现功能

- 欢迎进入
- 退出提示
- cd
- cd ~
- cd ../
- cd /
- cd +目录
- ls  多项功能
- echo
- cat
- pwd
- 重定向  >
- 重定向  >>
- 错误命令提示
- 错误路径提示

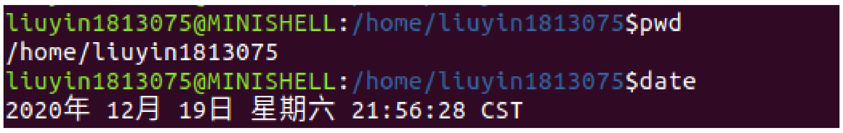
**操作截图：**

1. 程序启动，输出欢迎提示，设定所有地址启动 minishell 都预先返回 home/liuyin1813075



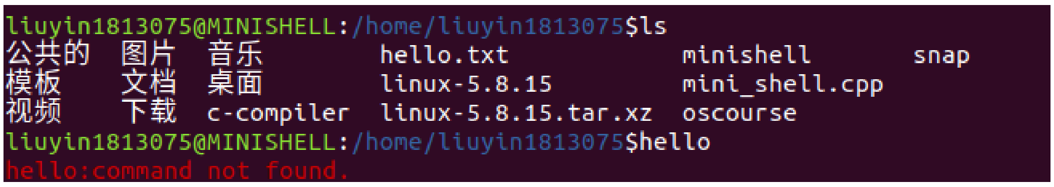
```
liuyin1813075@liuyin-VirtualBox:~/oscourse/course9$ g++ -g -o minishell mini_she
ll.cpp
liuyin1813075@liuyin-VirtualBox:~/oscourse/course9$ ./minishell
**************welcome to mini shell**************
liuyin1813075@MINISHELL:/home/liuyin1813075$
```

2.　pwd date 功能



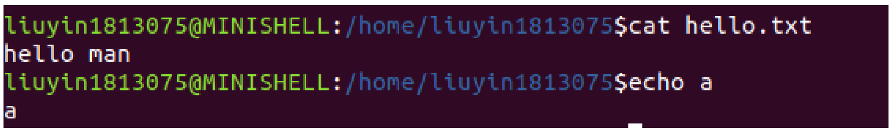
```
liuyin1813075@MINISHELL:/home/liuyin1813075$pwd
/home/liuyin1813075
liuyin1813075@MINISHELL:/home/liuyin1813075$date
2020年 12月 19日 星期六 21:56:28 CST
```

3.　ls 命令，无效操作命令的提示



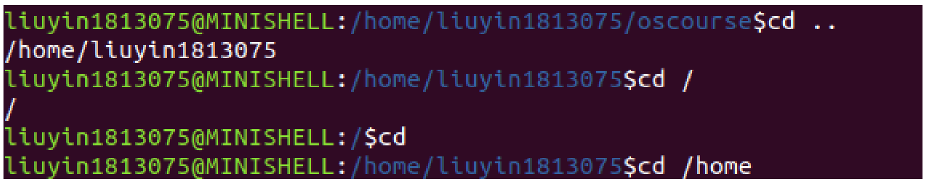
```
liuyin1813075@MINISHELL:/home/liuyin1813075$ls
公共的  图片  音乐        hello.txt            minishell       snap
模板    文档  桌面        linux-5.8.15         mini_shell.cpp
视频    下载  c-compiler  linux-5.8.15.tar.xz  oscourse
liuyin1813075@MINISHELL:/home/liuyin1813075$hello
hello:command not found.
```

4.　cat 命令查看文件并打印，echo 函数

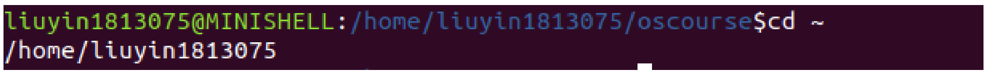


```
liuyin1813075@MINISHELL:/home/liuyin1813075$cat hello.txt
hello man
liuyin1813075@MINISHELL:/home/liuyin1813075$echo a
a
```

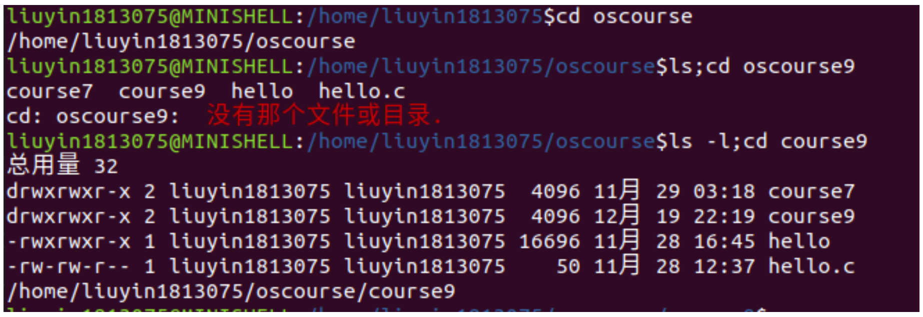
5.　cd 多种命令，ls 多命令



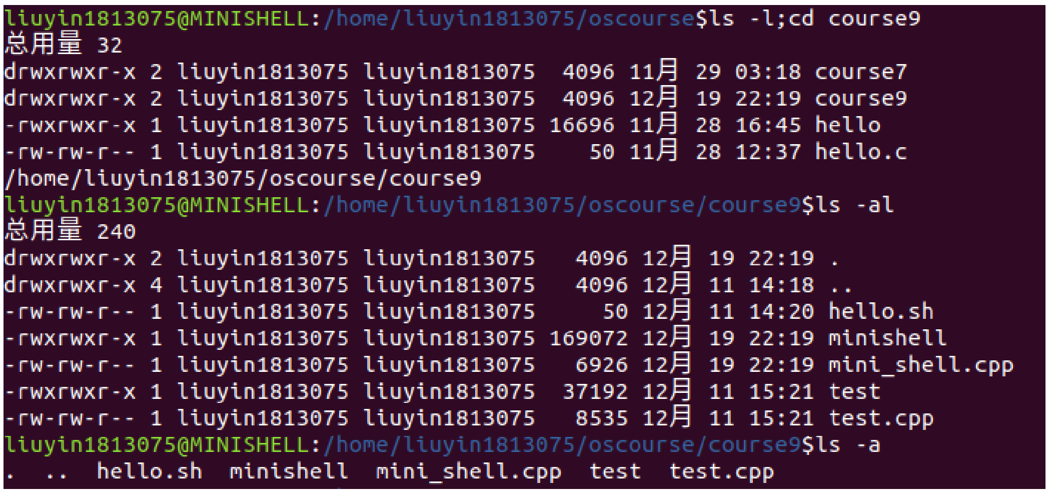
```
liuyin1813075@MINISHELL:/home/liuyin1813075/oscourse$cd ..
/home/liuyin1813075
liuyin1813075@MINISHELL:/home/liuyin1813075$cd /
/
liuyin1813075@MINISHELL:/$cd
liuyin1813075@MINISHELL:/home/liuyin1813075$cd /home
```



```
liuyin1813075@MINISHELL:/home/liuyin1813075/oscourse$cd ~
/home/liuyin1813075
```

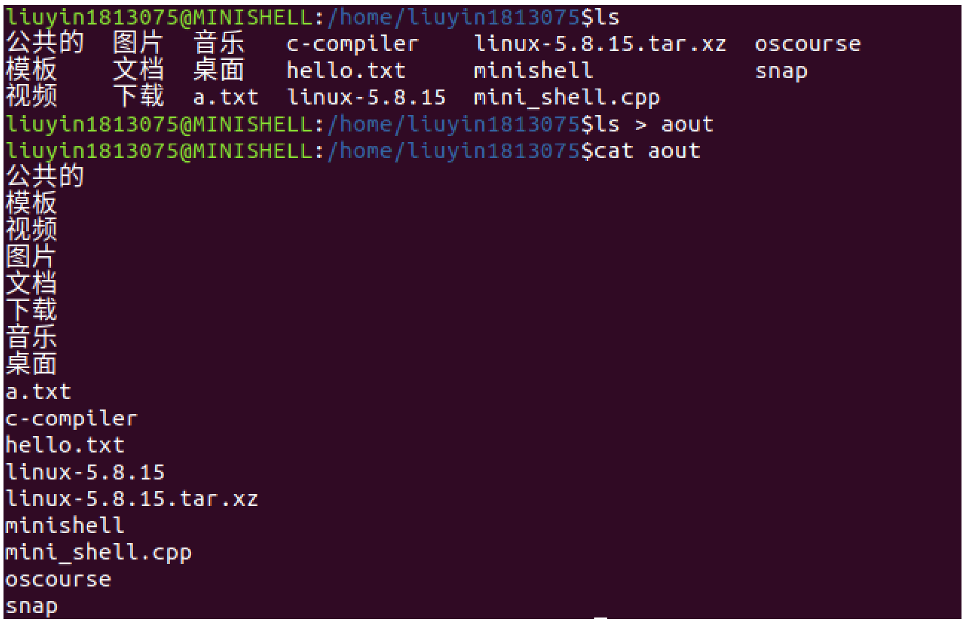


```
liuyin1813075@MINISHELL:/home/liuyin1813075$cd oscourse
/home/liuyin1813075/oscourse
liuyin1813075@MINISHELL:/home/liuyin1813075/oscourse$ls;cd oscourse9
course7  course9  hello  hello.c
cd: oscourse9:  没有那个文件或目录.
liuyin1813075@MINISHELL:/home/liuyin1813075/oscourse$ls -l;cd course9
总用量 32
drwxrwxr-x 2 liuyin1813075 liuyin1813075  4096 11月 29 03:18 course7
drwxrwxr-x 2 liuyin1813075 liuyin1813075  4096 12月 19 22:19 course9
-rwxrwxr-x 1 liuyin1813075 liuyin1813075 16696 11月 28 16:45 hello
-rw-rw-r-- 1 liuyin1813075 liuyin1813075    50 11月 28 12:37 hello.c
/home/liuyin1813075/oscourse/course9
```

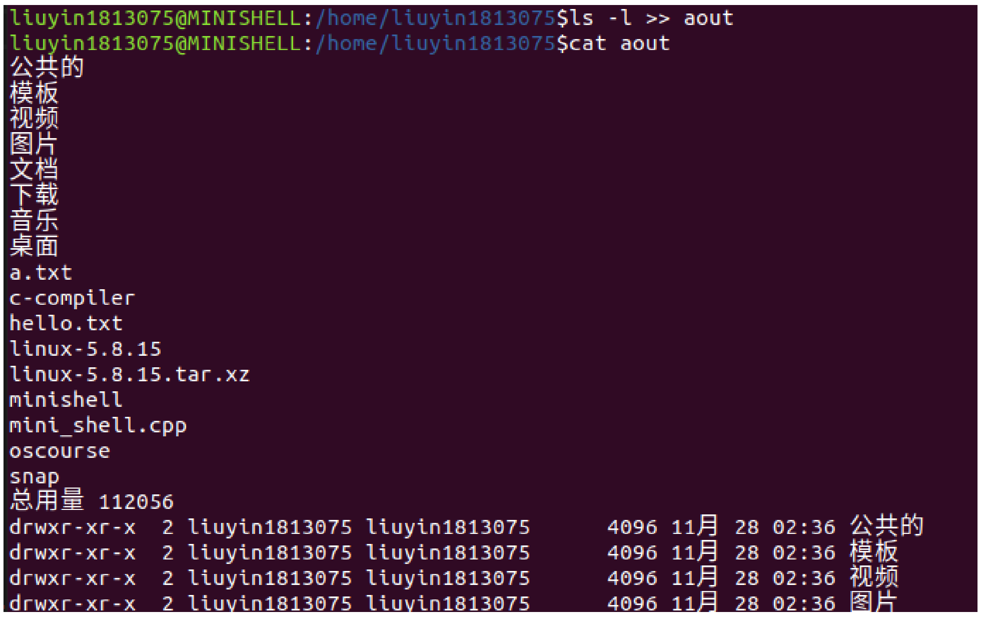


```
liuyin1813075@MINISHELL:/home/liuyin1813075/oscourse$ls -l;cd course9
总用量 32
drwxrwxr-x 2 liuyin1813075 liuyin1813075  4096 11月 29 03:18 course7
drwxrwxr-x 2 liuyin1813075 liuyin1813075  4096 12月 19 22:19 course9
-rwxrwxr-x 1 liuyin1813075 liuyin1813075 16696 11月 28 16:45 hello
-rw-rw-r-- 1 liuyin1813075 liuyin1813075    50 11月 28 12:37 hello.c
/home/liuyin1813075/oscourse/course9
liuyin1813075@MINISHELL:/home/liuyin1813075/oscourse/course9$ls -al
总用量 240
drwxrwxr-x 2 liuyin1813075 liuyin1813075   4096 12月 19 22:19 .
drwxrwxr-x 4 liuyin1813075 liuyin1813075   4096 12月 11 14:18 ..
-rw-rw-r-- 1 liuyin1813075 liuyin1813075     50 12月 11 14:20 hello.sh
-rwxrwxr-x 1 liuyin1813075 liuyin1813075 169072 12月 19 22:19 minishell
-rw-rw-r-- 1 liuyin1813075 liuyin1813075   6926 12月 19 22:19 mini_shell.cpp
-rwxrwxr-x 1 liuyin1813075 liuyin1813075  37192 12月 11 15:21 test
-rw-rw-r-- 1 liuyin1813075 liuyin1813075   8535 12月 11 15:21 test.cpp
liuyin1813075@MINISHELL:/home/liuyin1813075/oscourse/course9$ls -a
.  ..  hello.sh  minishell  mini_shell.cpp  test  test.cpp
```

6.　重定向 覆写和增加



```
liuyin1813075@MINISHELL:/home/liuyin1813075$ls
公共的  图片  音乐  c-compiler   linux-5.8.15.tar.xz  oscourse
模板    文档  桌面  hello.txt    minishell            snap
视频    下载  a.txt linux-5.8.15 mini_shell.cpp
liuyin1813075@MINISHELL:/home/liuyin1813075$ls > aout
liuyin1813075@MINISHELL:/home/liuyin1813075$cat aout
公共的
模板
视频
图片
文档
下载
音乐
桌面
a.txt
c-compiler
hello.txt
linux-5.8.15
linux-5.8.15.tar.xz
minishell
mini_shell.cpp
oscourse
snap
```



```
liuyin1813075@MINISHELL:/home/liuyin1813075$ls -l >> aout
liuyin1813075@MINISHELL:/home/liuyin1813075$cat aout
公共的
模板
视频
图片
文档
下载
音乐
桌面
a.txt
c-compiler
hello.txt
linux-5.8.15
linux-5.8.15.tar.xz
minishell
mini_shell.cpp
oscourse
snap
总用量 112056
drwxr-xr-x  2 liuyin1813075 liuyin1813075      4096 11月  28 02:36 公共的
drwxr-xr-x  2 liuyin1813075 liuyin1813075      4096 11月  28 02:36 模板
drwxr-xr-x  2 liuyin1813075 liuyin1813075      4096 11月  28 02:36 视频
drwxr-xr-x  2 liuyin1813075 liuyin1813075      4096 11月  28 02:36 图片
```

附：c++代码：

```cpp
#include <iostream>
#include <cstdio>
#include <string.h>
#include <unistd.h>
#include <pwd.h>
#include <libgen.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>
#include <vector>
#include <sstream>
#include <dirent.h>
#include <cstring>
```

```cpp
#include <fcntl.h>
using namespace std;
#define SUCCESS 0
#define ERROR -1
int reflag = 0;
char current_dir[100];
char user_dir[100];
vector<string> re;
int eval(vector<string> res);
int eval(vector<string> res)
{
    if (res[0] == "cd")
    {
        // 当前系统目录
        // char target_path[100];
        // getcwd(target_path, 100);
        // cout<<"now path"<<target_path<<endl;
        if (res.size() == 1)
        {
            strcpy(current_dir, user_dir);
        }
        else
        {
            const char *rest = res[1].c_str();
            // cout << "rest:" << rest << endl;
            // printf("\033[31mno such directory\n\033[0m");
            // printf("could open\n");
            if (res[1] == "/")
            {
                opendir(rest);
                strcpy(current_dir, rest);
                // cout<<"current / is"<<current_dir<<endl;
            }
            else if (res[1] == "..")
            {
                char *parent_dir = dirname(current_dir);
                strcpy(current_dir, parent_dir);
            }
            else if (res[1] == "~")
            {
                strcpy(current_dir, user_dir);
            }
            else
            {
                char target_path[100];
                cout << "current dit:" << current_dir << endl;
                if (strcmp(current_dir, "/") == 0)
                {
                    snprintf(target_path, 1024, "%s%s", current_dir, rest);
                }
                else
                {
                    snprintf(target_path, 1024, "%s/%s", current_dir, rest)
;
                }
                if (opendir(target_path) == NULL)
                {
                    cout << "cd: " << rest << ":";
                    printf("\033[31m  没有那个文件或目录.\n\033[0m");
                    return ERROR;
                }
                strcpy(current_dir, target_path);
            }
            cout << current_dir << endl;
            return ERROR;
        }
    }
    else if (res[0] == "pwd")
    {
        char buf[300];
        cout << current_dir << endl;
    }
    else if (res[0] == "ls")
    {
        int pid = fork(), wpid;
        int status;
        int count = 0;
        const char *rest = res[0].c_str();
        // printf(rest); //print ok

        if (pid == 0)
        {
            char *env[] = {0, NULL};
            for (int i = 0; i < res.size(); i++)
            {
                // cout<<"res"<<res[i]<<endl;
                if (res[i] == ">")
                {
                    // cout<<"check <"<<endl;
                    reflag = 1;
                    count = i;
                }
                if (res[i] == ">>")
                {
                    reflag = 2;
                    count = i;
                }
            }
            if (reflag != 0)
            {
                // cout<<"reflag!=0"<<endl;
                char **cmd_temp = new char *[count];

                for (int i = 0; i < count; i++)
                {
                    cmd_temp[i] = new char[500];
                    memset(cmd_temp[i], 0, sizeof(cmd_temp[i]));
                }
                for (int i = 0; i < count; i++)
                {
                    // cout << "now path" << res[i].c_str() << endl;
                    strcpy(cmd_temp[i], res[i].c_str());
                }
                cmd_temp[count] = current_dir;
                cmd_temp[count + 1] = NULL;
                // 标准输出重定向，将原本要写入标准输出 1 的数据写入新文件(fd)中
                int fd = 1;
                if (reflag == 1)
                    fd = open(res[count + 1].c_str(), O_CREAT | O_WRONLY |
O_TRUNC, 0664);
                else if (reflag == 2)
                    fd = open(res[count + 1].c_str(), O_CREAT | O_WRONLY |
O_APPEND, 0664);
                dup2(fd, 1);
                if (execvp(rest, cmd_temp) < 0)
                {
                    printf("\033[31m%s:command not found.\n\033[0m", res[0]
.c_str());
                }
            }
            else
            {
                // cout<<"reflag00"<<endl;
                char **cmd_temp = new char *[res.size() + 1];
                for (int i = 0; i < res.size(); i++)
                {
                    cmd_temp[i] = new char[500];
                    memset(cmd_temp[i], 0, sizeof(cmd_temp[i]));
                }
                for (int i = 0; i < res.size(); i++)
                {
                    // cout << "now path" << res[i].c_str() << endl;
                    strcpy(cmd_temp[i], res[i].c_str());
                }
                cmd_temp[res.size()] = current_dir;
                cmd_temp[res.size() + 1] = NULL;
                if (execvp(rest, cmd_temp) < 0)
                {
                    printf("\033[31m%s:command not found.\n\033[0m", res[0].c_str());
                }
            }
        }
        else if (pid > 0)
        {
            do
            {
                wpid = waitpid(pid, &status, WUNTRACED);
            } while (!WIFEXITED(status) && !WIFSIGNALED(status));
        }
    }
    else
    {
        int pid = fork(), wpid;
        int status;
        const char *rest = res[0].c_str();
        // printf(rest); //print ok

        if (pid == 0)
        {
            char **cmd_temp = new char *[res.size()];
            char *env[] = {0, NULL};
            for (int i = 0; i < res.size(); i++)
            {
                cmd_temp[i] = new char[500];
                memset(cmd_temp[i], 0, sizeof(cmd_temp[i]));
            }
            for (int i = 0; i < res.size(); i++)
            {
                strcpy(cmd_temp[i], res[i].c_str());
            }
            cmd_temp[res.size()] = NULL;
            if (execvp(rest, cmd_temp) < 0)
            {
                printf("\033[31m%s:command not found.\n\033[0m", res[0].c_s
tr());
            }
        }
        else if (pid > 0)
        {
            do
            {
                wpid = waitpid(pid, &status, WUNTRACED);
            } while (!WIFEXITED(status) && !WIFSIGNALED(status));
        }
    }
    return SUCCESS;
}
vector<string> split(const string &s, const string &seperator)
{
    vector<string> result;
    typedef string::size_type string_size;
    string_size i = 0;
    while (i != s.size())
    {
        int flag = 0;
        while (i != s.size() && flag == 0)
        {
            flag = 1;
            for (string_size x = 0; x < seperator.size(); ++x)
                if (s[i] == seperator[x])
                {
                    ++i;
                    flag = 0;
                    break;
                }
        }
        flag = 0;
        string_size j = i;
        while (j != s.size() && flag == 0)
        {
            for (string_size x = 0; x < seperator.size(); ++x)
                if (s[j] == seperator[x])
                {
                    flag = 1;
                    break;
                }
            if (flag == 0)
                ++j;
        }
        if (i != j)
        {
            result.push_back(s.substr(i, j - i));
            i = j;
        }
    }
    // 打印split 命令 √
    // for (int i = 0; i < result.size(); ++i)
    // {
    //     cout << i << ":" << result[i] << " " << endl;
    // }
    return result;
}
int main()
{
    string cmdstring;
    printf("\033[32m*************welcome to mini shell************* \n\033[0m");
    //test
    // printf("hello!\n");
    strcpy(current_dir, getpwuid(getuid())->pw_dir);
    strcpy(user_dir, getpwuid(getuid())->pw_dir);
    printf("\033[92m%s@MINISHELL\033[0m:\033[34m%s\033[0m$", getlogin(), current_dir);

    while (1)
    {
        for (int i = 0; i < re.size(); i++)
        {
            re[i].clear();
        }
        getline(cin, cmdstring); //input string with ' '
        string result;
        vector<string> v = split(cmdstring, ";"); //slipt command
        for (int i = 0; i < v.size(); i++)
        {
            re.clear();
            stringstream input2(v[i]); //string stream  initialize 不按照空格划分
            while (input2 >> result)
            {
                re.push_back(result);
            }
            if (result == "exit")
            {
                printf("\033[32m****************** mini shell exit******************\n\033[0m");
                return 0;
            }
            if (re.size())
            {
                // int a = re.size();
                // cout << "size:" << a << endl;
                eval(re);
            }
        }
        printf("\033[92m%s@MINISHELL\033[0m:\033[34m%s\033[0m$", getlogin(), current_dir);
    }
    return 0;
}
```