

选课序号：80 学号：2220213301 姓名：邹康

实验二：Linux 命令环境下 C/C++语言实践

一、实验目的

用 C/C++构造一个简单的 shell；
理解 shell 程序的功能；
学会 shell 的使用；
掌握基本的 makefile 方法。

二、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。

- 1) 浏览目录和文件的各种属性 ls（可以不支持参数）
- 2) 回显命令 echo
- 3) 显示文件内容 cat
- 4) 创建目录 mkdir
- 5) 删除文件 rm
- 6) 切换目录 cd
- 7) 显示当前目录 pwd
- 8) 文字统计 wc

基本任务 2：每一条命令单独对应一个源程序文件，不允许所有命令一个源文件。

基本任务 3：写一个 makefile 来管理这些源文件。

基本任务 4：写清楚 make 编译和运行的过程。

三、实验步骤

1. 编写 Shell 主程序：编写主程序，包括主要的 Shell 循环，用于读取用户输入和执行相应的命令。

```
void myshell_loop()
{
    char *line;
    char **args;
    int status;
    printf("-----欢迎来到我的 shell-----\n ");
    printf("-----可以通过 help 命令获取帮助\n ");
    do
    {
        getcwd(curpath, sizeof(curpath));
        getlogin_r(username, sizeof(username));
        gethostname(hostname, sizeof(hostname));
        printf("\e[32;1m%s@%s\e[0m:%s$ ", username, hostname, curpath); // 显示
为绿色

        line = myshell_readline();
        args = myshell_splitline(line);
```

```

        status = mysh_execute(args);

    } while (status);

}

```

```

int main(int argc, char *argv[])
{
    myshell_loop();
    return 0;
}

```

2. 实现命令解析：编写代码来解析用户输入的命令，包括分解命令、参数和选项。

```

char ** myshell_splitline(char * line)
{
    int buffer_size = MYSH_BUFFER_SIZE, position = 0;
    char **tokens = malloc(buffer_size * sizeof(char *));
    char *token;

    token = strtok(line, MYSH_TOK_DELIM);
    while(token != NULL)
    {
        tokens[position++] = token;
        //传递 NULL 作为第一个参数，以表示它应该继续使用相同的字符串（`line`）
        //来提取下一个令牌。
        token = strtok(NULL, MYSH_TOK_DELIM);
    }
    tokens[position] = NULL;
    return tokens;
}

```

3. 检查是否含有输出重定向，还是只有命令的执行

```

// 检查是否有输出重定向符号 ">"
for (i = 0; args[i] != NULL; i++) {
    if (strcmp(args[i], ">") == 0) {
        if (args[i + 1] == NULL) {
            perror("Missing output file name.\n");
            return 1;
        }
        output_redirect = i; //第几个分词是重定向符号
        for (int j = 0; j < output_redirect; j++)

```

```

        {
            args_a[j] = args[j];
        }

        output_filename = args[i + 1]; // 下一个参数是输出文件名
        break;
    }
}

```

4. 实现对命令的实现和执行：包括 ls, cat, help, exit, echo, touch, rm, mkdir, rmdir 等

//执行命令

```

int mysh_without_execute(char **args)
{
    if (args[0] == NULL) {
        return 1;
    } else if (strcmp(args[0], "help") == 0) {
        return mysh_help(args);
    } else if (strcmp(args[0], "exit") == 0) {
        return mysh_exit(args);
    } else if (strcmp(args[0], "ls") == 0) {
        return mysh_ls(args);
    } else if (strcmp(args[0], "cat") == 0) {
        return mysh_cat(args);
    } else if (strcmp(args[0], "echo") == 0) {
        return mysh_echo(args);
    } else if (strcmp(args[0], "mkdir") == 0) {
        return mysh_mkdir(args);
    } else if (strcmp(args[0], "rmdir") == 0) {
        return mysh_rmdir(args);
    } else if (strcmp(args[0], "touch") == 0) {
        return mysh_touch(args);
    } else if (strcmp(args[0], "rm") == 0) {
        return mysh_rm(args);
    } else if (strcmp(args[0], "pwd") == 0) {
        return mysh_pwd(args);
    } else if (strcmp(args[0], "wc") == 0) {
        return mysh_wc(args);
    } else if (strcmp(args[0], "cd") == 0) {
        return mysh_cd(args);
    } else {
        // 处理未知命令或错误
    }
}

```

```

    perror("Unknown command\n");
    return 1;
}

```

5. 各命令实现

1) Ls (-a, -l)

初始化 `ls_option` 数组，用于表示 `-l` 和 `-a` 选项是否被指定。默认情况下，它们都被设置为 0。

```

for ( i = 1; args[i] != NULL; i++) {
    // 如果参数不是以 "-" 开头，则它被认为是路径
    if (args[i][0] != '-') {
        path = args[i];
    }else{
        switch (args[i][1]) {
            case 'l':
                ls_option[ls_l] = 1;
                break;
            case 'a':
                ls_option[ls_a] = 1;
                break;
            default:
                perror("Unrecognized option\n");
                return 1;
                break;
        }
    }
}

```

调用 `list_files` 函数，传递 `path` 和 `ls_option` 数组作为参数，以列出文件和目录使用 `opendir` 函数打开指定路径，如果路径不存在，使用 `perror` 函数输出错误信息并返回。使用 `readdir` 函数遍历目录中的文件和子目录。

```

if ((dir = opendir(path)) == NULL) {
    perror("路径不存在\n");
    return;
}

while ((entry = readdir(dir)) != NULL) {
    if (entry->d_name[0] == '.') {
        if ( ls_option[ls_a] == 0 ) {
            continue; // 隐藏文件，如果不显示详情则跳过
        }
    }
}

```

使用 `lstat` 函数获取文件的详细信息,并编写一个函数 `mode_to_letters` 将用户权限转化为字符串，如果不需要显示详细信息，仅输出文件或目录的名称。

```

if (lstat(full_path, &file_info) == -1) {
    perror("lstat");
    continue;
}
//通过标志判断是否有 -l参数
if ( ls_option[ls_l] == 1) {
    // 显示详细信息
    char permissions[11];
    permissions[10] = '\0';
    mode_to_letters(file_info.st_mode, permissions);

    struct passwd *user_info = getpwuid(file_info.st_uid);
    struct group *group_info = getgrgid(file_info.st_gid);

    char time_buffer[80];
    strftime(time_buffer, sizeof(time_buffer), "%b %d %H:%M", localtime(&file_info.st_mtime));

    printf("%s %2ld %s %s %5ld %s %s\n", permissions, (long)file_info.st_nlink,
        user_info->pw_name, group_info->gr_name, (long)file_info.st_size, time_buffer, entry->d_name);
} else {
    printf("%s\t", entry->d_name);
}

```

2) cat

使用 `fopen` 函数打开文件,使用 `fgetc` 函数逐个字符读取文件内容,直到遇到文件末尾 (EOF),通过循环可以读取多个文件。但是不能不输入参数,等待用户输入。

```
// 从第二个参数开始,依次打开文件并输出内容
for (int i = 1; args[i] != NULL; i++) {
    FILE *file = fopen(args[i], "r");
    if (file == NULL) {
        perror("无法打开文件");
        return 1;
    }

    char ch;
    while ((ch = fgetc(file)) != EOF) {
        putchar(ch);
    }

    fclose(file);
}
```

3) cd

使用了 `chdir` 函数来尝试切换目录

```
if (chdir(args[1]) != 0)
    perror("error:路径未改变\n");
```

4) echo

初始化 `echo_option` 数组,用于表示 `-n` 和 `-e` 选项是否被指定。默认情况下,它们都被设置为 0。

```
switch (args[i][1]) {
    case 'n':
        echo_option[echo_n] = 1;
        break;
    case 'e':
        echo_option[echo_e] = 1;
        break;
    default:
        perror("Unrecognized option\n");
        return 1;
        break;
}
```

编写解释和打印包含转义字符的字符串的函数

```
for (int i = 0; input[i] != '\0'; i++) {
    if (input[i] == '\\') {
        i++;
        if (input[i] == '\\') {
            i++;
            if (input[i] == 'a') {
                output[output_index++] = '\a';
            } else if (input[i] == 'b') {
                output[output_index++] = '\b';
            } else if (input[i] == 'c') {
                break; // 禁止尾随的换行符
            } else if (input[i] == 'f') {
                output[output_index++] = '\f';
            } else if (input[i] == 'n') {
                output[output_index++] = '\n';
            } else if (input[i] == 'r') {
                output[output_index++] = '\r';
            } else if (input[i] == 't') {
                output[output_index++] = '\t';
            } else if (input[i] == 'v') {
                output[output_index++] = '\v';
            } else if (input[i] == '\\') {
                output[output_index++] = '\\'; // 反斜线
            } else if (input[i] == 'x') {
                // 处理 \xNN 形式的16进制转义序列
                i++; // 跳过 'x'
                char hex[3] = {input[i], input[i + 1], '\0'};
                output[output_index++] = (char)strtol(hex, NULL, 16);
                i++;
            }
        }
    } else {
        output[output_index++] = input[i];
    }
}
```

```
} else {
    i--;
    output[output_index++] = input[i];
}
} else {
    output[output_index++] = input[i];
}
}
```

如果没有-n 选项，输出换行符以结束行

```
if (!echo_option[echo_n]) {  
    printf("\n"); // 如果没有 -n 参数，添加回车符  
}
```

5) pwd

使用 getcwd 函数来获取当前工作目录的路径,获取成功 printf 函数来输出当前工作目录的路径。

```
if(getcwd(curpath,sizeof(curpath)) != NULL){  
    printf("%s\n",curpath);  
}else{  
    perror("mysh_pwd");  
}
```

6) mkdir

使用 mkdir 函数来尝试创建目录。默认情况下，使用 0777 权限，可以创建多级目录。

```
// 从第二个参数开始，依次创建目录  
for (int i = 1; args[i] != NULL; i++) {  
    if (mkdir(args[i], 0777) != 0) {  
        perror("创建失败,文件已经存在\n");  
        return 1;  
    }  
}
```

7) rmdir

使用 rmdir 函数来尝试删除目录

```
// 从第二个参数开始，依次删除目录  
for (int i = 1; args[i] != NULL; i++) {  
    if (rmdir(args[i]) != 0) {  
        perror("mysh_rmdir");  
        return 1;  
    }  
}
```

8) touch

使用 open 函数来尝试创建文件，给默认权限 0666.

```
// 从第二个参数开始，依次创建文件  
for (int i = 1; args[i] != NULL; i++) {  
    int fd = open(args[i], O_CREAT | O_WRONLY, 0666);  
    if (fd == -1) {  
        perror("mysh_touch");  
        return 1;  
    }  
    close(fd);  
}  
  
return 1;  
}
```

9) rm

使用 unlink 函数来尝试删除文件

```
// 从第二个参数开始，依次删除文件
for (int i = 1; args[i] != NULL; i++) {
    if (unlink(args[i]) != 0) {
        perror("mysh_rm");
        return 1;
    }
}
```

10) wc (-c,-l,-w)

检查命令行参数是否以-开头，如果是，则根据参数设置相应的标志给 wc_option。

```
switch (args[i][1]) {
    case 'c':
        wc_option[wc_c] = 1;
        break;
    case 'l':
        wc_option[wc_l] = 1;
        break;
    case 'w':
        wc_option[wc_w] = 1;
        break;
    default:
        perror("Unrecognized option\n");
        return 1;
        break;
}
```

对文件内容进行处理，使用 fopen 打开文件，初始化字符数、行数、单词数和一个标志变量 in_word，以便跟踪当前是否在一个单词内。在循环中，逐字符读取文件内容并统计字符数、行数和单词数。最后，关闭文件，并在根据选项打印统计结果

```
int lines = 0;
int words = 0;
int characters = 0;
int in_word = 0;
char ch;

while ((ch = fgetc(file)) != EOF) {
    characters++;
    if (ch == '\n') {
        lines++;
    }
    if ((ch == ' ' || ch == '\t' || ch == '\n')) {
        if (in_word) {
            words++;
            in_word = 0;
        }
    } else {
        in_word = 1;
    }
}
```

6. 编写帮助文档：创建帮助文档，以解释每个命令的使用方法和选项，以及每个命令的注意事项。

```
printf("Welcome to Help of My Shell !\n");
printf("-----\n");
printf("Command List as Following\n");
printf("1) 查看命令使用 help [command]\n"
"2) 回显命令 echo\n"
"3) 显示文件内容 cat\n"
"4) 创建目录 mkdir\n"
"5) 删除目录 rmdir\n"
"6) 创建文件 touch\n"
"7) 删除文件 rm\n"
"8) 切换目录 cd\n"
"9) 显示当前目录 pwd\n"
"10) 文字统计 wc\n"
"11) 结束程序 exit\n"
"12) 浏览目录和文件的各种属性 ls\n"
"13) 输出重定向的使用 cdx\n");
```

当输入是命令帮助，就要显示相关命令文档

使用 `snprintf` 函数构建帮助文档的路径。帮助文档以 `help/[command_name].txt` 的形式存储

```
char command_name[100]; // 存储命令帮助路径
snprintf(command_name, sizeof(command_name), "help/%s.txt", args[1])
```

使用 `fork` 函数创建一个子进程。在子进程中，将执行分页器来显示帮助文档。分页器使用 `less`。使用 `popen` 函数创建一个管道，以便将帮助文档传递给分页器。从帮助文档文件中逐行读取内容，并将每行内容写入管道，以便在分页器中显示。

```
if (pid == 0) {
    // 子进程中执行分页器
    char *pager = "less"; // 使用 less 分页器
    char *pager_args[] = {pager, NULL};
```



```

// 通过管道将帮助文档传递给分页器
FILE *pipe = popen(pager, "w");
char line[256];
while (fgets(line, sizeof(line), file)) {
    fputs(line, pipe);
}
fclose(file);
pclose(pipe);

// 子进程退出
exit(0);
} else if (pid > 0) {
    // 父进程等待子进程退出
    wait(NULL);
} else {
    perror("fork");
}
} else {
    printf("No help available for '%s'\n", args[1]);
}
}

```

7. 通过 makefile 管理源文件

CC = gcc: 这行代码定义了 CC 变量，指定了使用的 C 编译器。

target = 3301_mysh: 这行定义了 target 变量，指定了要生成的可执行文件的名称 3301_mysh。

src = \$(wildcard *.c): 这行使用 wildcard 函数查找当前目录中的所有 .C 文件，并将它们赋值给 src 变量。

objects = \$(patsubst %.c, %.o, \$(src)): 这行生成了一个对象文件列表，通过将源文件的扩展名从 .c 替换为 .o 来实现。这些对象文件被赋值给 objects 变量。

\$(target): \$(objects): 这行规定 target 目标文件依赖于 objects (.c 文件)

\$(CC) -std=c99 -o \$(target) \$(objects): 这行是构建命令，使用 CC 变量 (GCC) 编译 objects 中的 C 源文件，使用 C99 标准生成 target 可执行文件。

%.o: %.c: C 源文件 (%.c) 编译为对象文件 (%.o)。它使用 CC 变量和 C99 标准来编译每个源文件为一个对象文件。

.PHONY: clean: 这行规定了一个“phony”目标，名为 clean，用于执行清理操作。在这种情况下，它用于清理生成的可执行文件和对象文件。

clean: 此规则指定了删除 target 和所有 objects 文件的命令，用于清理项目。

```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
CC = gcc
target = 3301_mysh
src = $(wildcard *.c)
objects = $(patsubst %.c, %.o, $(src))
$(target): $(objects)
    $(CC) -std=c99 -o $(target) $(objects)
%.o: %.c
    $(CC) -std=c99 -c $<

.PHONY: clean
clean:
    rm $(target) $(objects)

```

8. make 编译，生成可执行文件 3301_mysh

```

[zaki01@ZAKI1 lab2] $ ls
3301_mysh  mysh_cat.c  myshell.c  mysh_ls.c  mysh_rm.c  mysh_wc.c
a.txt      mysh_cat.o  myshell.o  mysh_ls.o  mysh_rmdir.c  mysh_wc.o
b.txt      mysh_cd.c  mysh_exit.c  mysh_mkdir.c  mysh_rmdir.o
help       mysh_cd.o  mysh_exit.o  mysh_mkdir.o  mysh_rm.o
m          mysh_echo.c  mysh_help.c  mysh_pwd.c  mysh_touch.c
makefile   mysh_echo.o  mysh_help.o  mysh_pwd.o  mysh_touch.o

```

四、实验结果:

1. 运行 make 进行编译 --> ./3301_mysh

```

[zaki01@ZAKI1 lab2] $ ./3301_mysh
----- 欢迎来到我的 shell-----
----- 可以通过help命令获取帮助
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$

```

2. 查看帮助文档 help (用户手册)，里面包含用户注意事项和命令的使用方法:

```

zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ help
Welcome to Help of My Shell !

```

```

-----
Command List as Following
1) 查看命令使用 help [command]
2) 回显命令 echo
3) 显示文件内容 cat
4) 创建目录 mkdir
5) 删除目录 rmdir
6) 创建文件 touch
7) 删除文件 rm
8) 切换目录 cd
9) 显示当前目录 pwd
10) 文字统计 wc
11) 结束程序 exit
12) 浏览目录和文件的各种属性 ls
13) 输出重定向的使用 cdx

```

3. 查看命令使用帮助 help ls/.... (按 q 退出查看)

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
MYSHELL Commands Manual
ls
NAME
ls - 列出目录中的文件和子目录
SYNOPSIS
ls [OPTION]... [DIRECTORY]...
DESCRIPTION
ls 命令用于列出指定目录中的文件和子目录。
OPTIONS
-l 显示文件的详细信息，包括权限、所有者、组、大小和修改时间。
-a 显示隐藏文件，包括以点开头的文件和目录。
DIRECTORY
要列出文件和子目录的目录。如果未指定目录，默认为当前目录，只能读取一个目录不能读取一个文件。
EXAMPLES
:

4. 执行 ls 命令 (ls, ls-l, ls-a)

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls
3301_mysh      a.txt  b.txt  help    m      makefile      myshell.c      myshell.o
mysh_cat.c     mysh_cat.o  mysh_cd.c  mysh_cd.o  mysh_echo.c  mysh_echo.o
mysh_exit.c    mysh_exit.o  mysh_help.c  mysh_help.o  mysh_ls.c    mysh_ls.o
mysh_mkdir.c   mysh_mkdir.o  mysh_pwd.c  mysh_pwd.o  mysh_rm.c    mysh_rm.o
mysh_rmdir.c   mysh_rmdir.o  mysh_touch.c  mysh_touch.o  mysh_wc.c    mysh_wc.o
```

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls -a
.      ..      .makefile.swp  .vscode  3301_mysh      a.txt  b.txt  help    m      makefile
myshell.c  myshell.o  mysh_cat.c  mysh_cat.o  mysh_cd.c  mysh_cd.o  m
ysh_echo.c  mysh_echo.o  mysh_exit.c  mysh_exit.o  mysh_help.c  mysh_help.o  m
ysh_ls.c    mysh_ls.o  mysh_mkdir.c  mysh_mkdir.o  mysh_pwd.c  mysh_pwd.o  m
ysh_rm.c    mysh_rm.o  mysh_rmdir.c  mysh_rmdir.o  mysh_touch.c  mysh_touch.o  m
ysh_wc.c    mysh_wc.o
```

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls -l
-rwxrwxrwx 1 root root 24192 Oct 11 22: 40 3301_mysh
-rwxrwxrwx 1 root root 4 Oct 11 21: 43 a.txt
-rwxrwxrwx 1 root root 0 Oct 11 22: 00 b.txt
drwxrwxrwx 1 root root 4096 Oct 11 22: 21 help
drwxrwxrwx 1 root root 0 Oct 11 20: 41 m
-rwxrwxrwx 1 root root 230 Oct 11 10: 47 makefile
-rwxrwxrwx 1 root root 5371 Oct 14 13: 12 myshell.c
-rwxrwxrwx 1 root root 7168 Oct 11 22: 30 myshell.o
-rwxrwxrwx 1 root root 582 Oct 11 22: 30 mysh_cat.c
-rwxrwxrwx 1 root root 2040 Oct 11 22: 30 mysh_cat.o
-rwxrwxrwx 1 root root 383 Oct 11 22: 30 mysh_cd.c
-rwxrwxrwx 1 root root 1720 Oct 11 22: 30 mysh_cd.o
-rwxrwxrwx 1 root root 3116 Oct 11 22: 30 mysh_echo.c
-rwxrwxrwx 1 root root 3136 Oct 11 22: 30 mysh_echo.o
-rwxrwxrwx 1 root root 171 Oct 10 16: 20 mysh_exit.c
-rwxrwxrwx 1 root root 1240 Oct 11 22: 30 mysh_exit.o
-rwxrwxrwx 1 root root 1908 Oct 11 22: 40 mysh_help.c
-rwxrwxrwx 1 root root 3272 Oct 11 22: 40 mysh_help.o
-rwxrwxrwx 1 root root 3485 Oct 11 22: 30 mysh_ls.c
```

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls -l -a
drwxrwxrwx 1 root root 4096 Oct 11 22: 40 .
drwxrwxrwx 1 root root 0 Oct 11 20: 38 ..
-rwxrwxrwx 1 root root 12288 Oct 10 14: 45 .makefile.swp
drwxrwxrwx 1 root root 0 Oct 10 20: 44 .vscode
-rwxrwxrwx 1 root root 24192 Oct 11 22: 40 3301_mysh
-rwxrwxrwx 1 root root 4 Oct 11 21: 43 a.txt
-rwxrwxrwx 1 root root 0 Oct 11 22: 00 b.txt
drwxrwxrwx 1 root root 4096 Oct 11 22: 21 help
drwxrwxrwx 1 root root 0 Oct 11 20: 41 m
```


4.echo 命令(echo -n /-e):

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ echo adsf\\taf\\f\\  
adsf\\taf\\f\\  
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ echo -e adsf\\taf\\fs\\  
adsf    affs\\  
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ echo -n adaf  
adafzaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ echo -n -e adsf\\nfadf  
adsfnfadfzaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$
```

5.cat 命令:

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ cat mysh_cd.c  
#include <unistd.h>  
#include <stdlib.h>  
#include <stdio.h>  
#include <string.h>  
#include <sys/wait.h>  
#include <sys/types.h>  
  
int mysh_cd(char **args)  
{  
    if(args[1] == NULL)  
    {  
        perror("参数错误,使用help命令查看帮助\\n");  
        return 1;  
    }  
    else  
    {  
        if(chdir(args[1]) != 0)  
            perror("error: 路径未改变\\n");  
    }  
    return 1;  
}
```

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ cat mysh_cd.c a.txt  
#include <unistd.h>  
#include <stdlib.h>  
#include <stdio.h>  
#include <string.h>  
#include <sys/wait.h>  
#include <sys/types.h>  
  
int mysh_cd(char **args)  
{  
    if(args[1] == NULL)  
    {  
        perror("参数错误,使用help命令查看帮助\\n");  
        return 1;  
    }  
    else  
    {  
        if(chdir(args[1]) != 0)  
            perror("error: 路径未改变\\n");  
    }  
    return 1;  
}  
- 1
```



6.mkdir 命令:


```

zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ mkdir ad
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls
3301_mysh      a.txt  ad      b.txt  help  m      makefile      myshell.c      myshell.
o      mysh_cat.c      mysh_cat.o      mysh_cd.c      mysh_cd.o      mysh_echo.c      mysh_echo
o.o      mysh_exit.c      mysh_exit.o      mysh_help.c      mysh_help.o      mysh_ls.c      mysh_ls.
o      mysh_mkdir.c      mysh_mkdir.o      mysh_pwd.c      mysh_pwd.o      mysh_rm.c      mysh_rm.
o      mysh_rmdir.c      mysh_rmdir.o      mysh_touch.c      mysh_touch.o      mysh_wc.c      mysh_wc.
o

```

7.rmdir 命令:

```

zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls
3301_mysh      a.txt  ad      b.txt  help  m      makefile      myshell.c      myshell.
o      mysh_cat.c      mysh_cat.o      mysh_cd.c      mysh_cd.o      mysh_echo.c      mysh_echo
o.o      mysh_exit.c      mysh_exit.o      mysh_help.c      mysh_help.o      mysh_ls.c      mysh_ls.
o      mysh_mkdir.c      mysh_mkdir.o      mysh_pwd.c      mysh_pwd.o      mysh_rm.c      mysh_rm.
o      mysh_rmdir.c      mysh_rmdir.o      mysh_touch.c      mysh_touch.o      mysh_wc.c      mysh_wc.
o
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ rmdir ad
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls
3301_mysh      a.txt  b.txt  help  m      makefile      myshell.c      myshell.o      m
ysh_cat.c      mysh_cat.o      mysh_cd.c      mysh_cd.o      mysh_echo.c      mysh_echo.o      m
ysh_exit.c      mysh_exit.o      mysh_help.c      mysh_help.o      mysh_ls.c      mysh_ls.o      m
ysh_mkdir.c      mysh_mkdir.o      mysh_pwd.c      mysh_pwd.o      mysh_rm.c      mysh_rm.o      m
ysh_rmdir.c      mysh_rmdir.o      mysh_touch.c      mysh_touch.o      mysh_wc.c      mysh_wc.o

```

8.touch 命令:

```

zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ touch c.txt
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls
3301_mysh      a.txt  b.txt  c.txt  help  m      makefile      myshell.c      myshell.
o      mysh_cat.c      mysh_cat.o      mysh_cd.c      mysh_cd.o      mysh_echo.c      mysh_echo
o.o      mysh_exit.c      mysh_exit.o      mysh_help.c      mysh_help.o      mysh_ls.c      mysh_ls.
o      mysh_mkdir.c      mysh_mkdir.o      mysh_pwd.c      mysh_pwd.o      mysh_rm.c      mysh_rm.
o      mysh_rmdir.c      mysh_rmdir.o      mysh_touch.c      mysh_touch.o      mysh_wc.c      mysh_wc.
o

```

9.rm 命令:

```

zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls
3301_mysh      a.txt  b.txt  c.txt  help  m      makefile      myshell.c      myshell.
o      mysh_cat.c      mysh_cat.o      mysh_cd.c      mysh_cd.o      mysh_echo.c      mysh_echo
o.o      mysh_exit.c      mysh_exit.o      mysh_help.c      mysh_help.o      mysh_ls.c      mysh_ls.
o      mysh_mkdir.c      mysh_mkdir.o      mysh_pwd.c      mysh_pwd.o      mysh_rm.c      mysh_rm.
o      mysh_rmdir.c      mysh_rmdir.o      mysh_touch.c      mysh_touch.o      mysh_wc.c      mysh_wc.
o
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ rm c.txt b.txt
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls
3301_mysh      a.txt  help  m      makefile      myshell.c      myshell.o      mysh_cat
.c      mysh_cat.o      mysh_cd.c      mysh_cd.o      mysh_echo.c      mysh_echo.o      mysh_exi
t.c      mysh_exit.o      mysh_help.c      mysh_help.o      mysh_ls.c      mysh_ls.o      mysh_mkd
ir.c      mysh_mkdir.o      mysh_pwd.c      mysh_pwd.o      mysh_rm.c      mysh_rm.o      mysh_rmd
ir.c      mysh_rmdir.o      mysh_touch.c      mysh_touch.o      mysh_wc.c      mysh_wc.o

```

10.cd 命令:

```

zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ cd m
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2/m$ ls
b      b.c      b.o

```

11.pwd 命令:

```

zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2/m$ pwd
/mnt/hgfs/linux Myshare/lab2/m

```

12.wc 命令(wc -c /-l /-w):

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ wc a.txt
Characters: 327 Lines: 1 Words: 31 a.txt
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ wc -l a.txt
Lines: 1 a.txt
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ wc -w a.txt
Words: 31 a.txt
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ wc -c a.txt
Characters: 327 a.txt
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ wc -l -c a.txt
Characters: 327 Lines: 1 a.txt
```

13.实现输出重定向>:

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ cat a.txt
3301_mysh      a.txt      help      m      makefile      myshell.c      myshell.o      mysh_cat
.c      mysh_cat.o      mysh_cd.c      mysh_cd.o      mysh_echo.c      mysh_echo.o      mysh_exit
t.c      mysh_exit.o      mysh_help.c      mysh_help.o      mysh_ls.c      mysh_ls.o      mysh_mkd
ir.c      mysh_mkdir.o      mysh_pwd.c      mysh_pwd.o      mysh_rm.c      mysh_rm.o      mysh_rmd
ir.c      mysh_rmdir.o      mysh_touch.c      mysh_touch.o      mysh_wc.c      mysh_wc.o
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ ls -l > a.txt
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ cat a.txt
-rwxrwxrwx 1 root root 24192 Oct 11 22:40 3301_mysh
-rwxrwxrwx 1 root root 53 Oct 14 13:49 a.txt
drwxrwxrwx 1 root root 4096 Oct 14 13:45 help
drwxrwxrwx 1 root root 0 Oct 11 20:41 m
-rwxrwxrwx 1 root root 230 Oct 11 10:47 makefile
-rwxrwxrwx 1 root root 5371 Oct 14 13:12 myshell.c
-rwxrwxrwx 1 root root 7168 Oct 11 22:30 myshell.o
```

14.退出 shell:

```
zaki01@ZAKI1: /mnt/hgfs/linux Myshare/lab2$ exit
[zaki01@ZAKI1 lab2]$
```

15.使用 make clean 清除.o 文件和目标文件

五、实验分析

本次实验的主要目标是通过编程方式深入理解基本命令的实现原理和工作机制。通过实验，我成功创建了一个包含多个基本命令的自定义 Shell 程序，并编写了相关代码以支持这些命令的功能。这包括内置命令（如`cd`、`echo`、`exit`、`help`）以及外部命令的执行。

通过使用 Makefile，我能够有效地管理多个源文件并将它们编译成一个可执行的 Shell 程序。这提高了我对 Linux 编译和构建管理的理解，并让我能够更好地组织和维护我的代码。

总的来说，这次实验增强了我对 Linux Shell 命令和编程的理解，通过自己编写 Shell 程序，我深入了解了各种命令的功能和使用方法。这将有助于我更好地理解 and 利用 Linux 系统中的命令行工具，并提升了我的编程技能和 Linux 系统管理能力。