

**软件学院学生实验报告**

**实验课程名称：** 计算机操作系统 **教师：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | **命令行解释程序的模拟** | | | **实验成绩** |  |
| **学生姓名** |  | **学 号** | **029** | **年级专业班级** |  |
| **小组成员** |  | | | **实验日期** | **17年9月26日** |

1. **实验目的**

命令行解释程序的模拟

**二、实验软硬件环境**

虚拟机VMware Workstation Pro、Ubuntu操作系统

1. **实验内容及要求**

（一）使用高级语言编写一个命令行解释程序

（二）实现至少三种shell命令

1. **实验记录**

**（一）程序概述**

1. **简要概述**

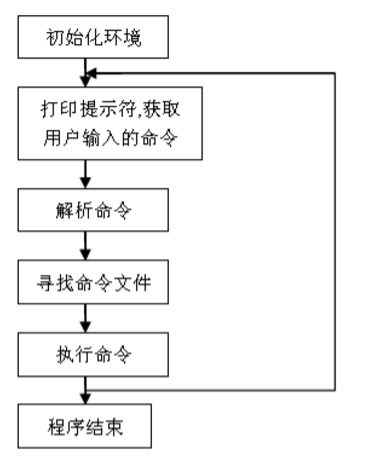
本实验完成了一个shell命令解释器，实现了shell的解释命令功能

**2.实现的功能**

|  |  |  |
| --- | --- | --- |
| 函数 | 功能 | 说明 |
| 实现了void mydir() | 列出目录和文件功能 | 具有dir 的功能 |
| 实现了void mycd() | 改变当前文件夹功能 | 具有cd  的功能 |
| 实现了void mycopy() | 同一个文件夹内复制文件功能 | 具有cp的功能 |
| 实现了void mydel() | 删除空目录功能（当删除的目录内又文件时会失败） | 具有del 的功能 |
| 实现了void mywd() | 显示当前目录名的功能 | 具有pwd 的功能 |
| 实现了 void exit() | 在实践中必须用到的退出程序的功能 |  |

**3.主要程序流程图**

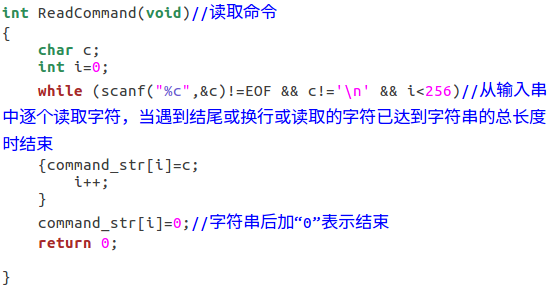
命令的分析执行过程包括：初始化环境，打印提示符，获取用户输入命令，解析命令，寻找命令文件和执行命令，如图

**（二）主要功能实现方法和系统调用**

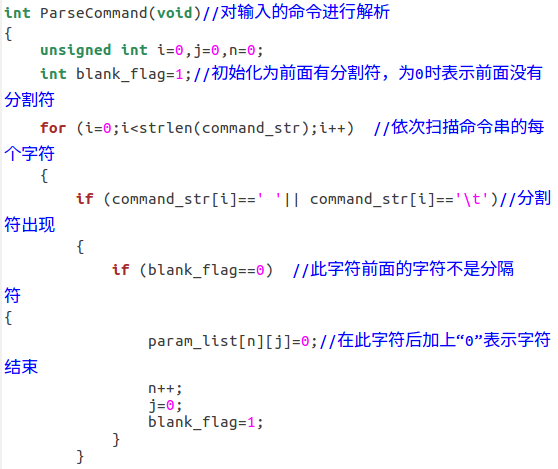
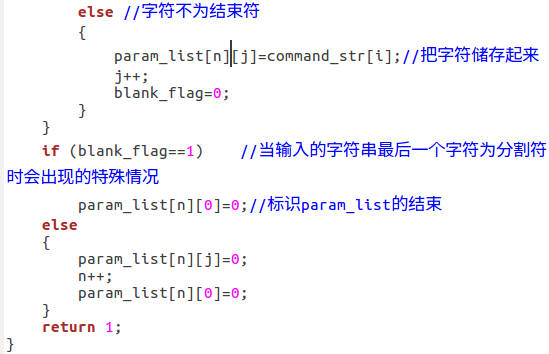
1. **系统调用**

总结起来用到的系统调用主要有：

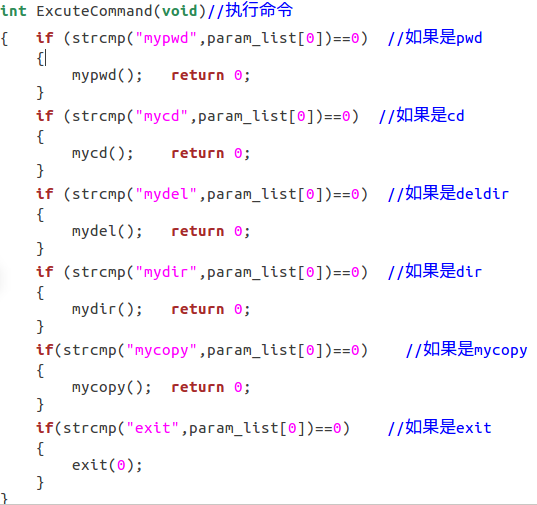
open();close();chdir();getcwd();exit();opendir（）；readdir();等等

**2.读取命令**

**3.解析命令**



**4.执行命令**



1. **详细设计**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <dirent.h>

#include <sys/wait.h>

char command\_str[256]; //存储输入的命令串

char param\_list[20][256];//储存分析好的命令串

int ReadCommand(void);

int ParseCommand(void);

int ExcuteCommand(void);

int mycopy(void);

int mypwd(void);

int mycd(void);

int mydel(void);

int mydir(void);

int exit(void);

**int ReadCommand(void)//读取命令**

{

char c;

int i=0;

while (scanf("%c",&c)!=EOF && c!='\n' && i<256)//从输入串中逐个读取字符，当遇到结尾或换行或读取的字符已达到字符串的总长度时结束

{command\_str[i]=c;

i++;

}

command\_str[i]=0;//字符串后加“0”表示结束

return 0;

}

**int ParseCommand(void)//对输入的命令进行解析**

{

unsigned int i=0,j=0,n=0;

int blank\_flag=1;//初始化为前面有分割符，为0时表示前面没有分割符

for (i=0;i<strlen(command\_str);i++) //依次扫描命令串的每个字符

{

if (command\_str[i]==' '|| command\_str[i]=='\t')//分割符出现

{

if (blank\_flag==0) //此字符前面的字符不是分隔符

{

param\_list[n][j]=0;//在此字符后加上“0”表示字符结束

n++;

j=0;

blank\_flag=1;

}

}

else //字符不为结束符

{

param\_list[n][j]=command\_str[i];//把字符储存起来

j++;

blank\_flag=0;

}

}

if (blank\_flag==1) //当输入的字符串最后一个字符为分割符时会出现的特殊情况

param\_list[n][0]=0;//标识param\_list的结束

else

{

param\_list[n][j]=0;

n++;

param\_list[n][0]=0;

}

return 1;

}

**int ExcuteCommand(void)//执行命令**

{ if (strcmp("mypwd",param\_list[0])==0) //如果是pwd

{

mypwd(); return 0;

}

if (strcmp("mycd",param\_list[0])==0) //如果是cd

{

mycd(); return 0;

}

if (strcmp("mydel",param\_list[0])==0) //如果是deldir

{

mydel(); return 0;

}

if (strcmp("mydir",param\_list[0])==0) //如果是dir

{

mydir(); return 0;

}

if(strcmp("mycopy",param\_list[0])==0) //如果是mycopy

{

mycopy(); return 0;

}

if(strcmp("exit",param\_list[0])==0) //如果是exit

{

exit(0);

}

}

**int mycopy(void)//复制文件**

{

char sourcepath[255];//被复制文件路径，旧的

char aimpath[255];//复制的文件路径，新的

FILE \*newfp;// 复制的文件指针，新的

FILE \*oldfp;//被复制的文件指针，旧的

char ch; //复制内容时用的载体

getcwd(sourcepath,255);// 将当前工作目录的绝对路径复制到参数sourcepath所指的内存空间中,255为soucerpath的空间大小

getcwd(aimpath,255);//同上

strcat(sourcepath,"/");//将字符串“/”添加到sourcepath结尾处

strcat(sourcepath,param\_list[1]);//将字符串（即被复制的文件名）添加到sourcepath结尾处

strcat(aimpath,"/");

strcat(aimpath,param\_list[2]); //将字符串（即需要复制的文件名）添加到aimpath结尾处

if((oldfp=fopen(sourcepath,"r"))==NULL)//以只读方式打开被复制文件，如果打开失败进行提示

{

printf("不能打开指定文件!\n");

}

if((newfp=fopen(aimpath,"w"))==NULL)//以只写方式打开被复制文件，如果打开失败进行提示

{

printf("新建文件失败!\n");

}

while((ch=fgetc(oldfp))!=EOF)//从被复制文件中逐个读取字符，如果读到文件末尾结束

{

fputc(ch,newfp);//把ch里的字符储存在新的文件里

}

fclose(oldfp);//关闭文件

fclose(newfp);

printf("成功的把文件从 %s 复制到 %s !\n",sourcepath,aimpath);

}

int mypwd(void)//显示当前目录

{

char buf[256];

getcwd(buf,sizeof(buf));//获取当前工作目录的绝对路径

printf("%s\n",buf);//输入

return 0;

}

**int mycd(void)//改变当前文件夹**

{

if (strcmp("",param\_list[1])==0)// cd后不带参数

return -1;

if (chdir(param\_list[1])==-1) //改变目录失败

{

printf("目录不存在!\n");

return -1;

}

return 0;

}

int mydel(void)//deldir命令

{

if (strcmp("",param\_list[1])==0) //命令不带参数

{

printf("命令参数错误!\n");

return -1;

}

if (rmdir(param\_list[1])==-1) //删除目录失败

{

printf("删除目录失败!\n");

return -1;

}

else

printf("删除目录成功!\n");

return 0;

}

**int mydir(void)//dir命令**

{

DIR\* dirp;

struct dirent \* ent;

if (strcmp("",param\_list[1])==0)//命令不带参数

dirp=opendir(".");//打开当前工作目录

else

dirp=opendir(param\_list[1]);//打开目录

if (dirp==NULL) //如果打开目录失败

{

printf("打开目录失败!\n");

return -1;

}

while ((ent=readdir(dirp))!=NULL) //依次读取目录的信息

{

if (ent->d\_name[0]!='.')

printf("%s\t",ent->d\_name);//%s表示数据输出类型为字符串

}

printf("\n");

closedir(dirp);//关闭目录

return 0;

}

**int main(int argc,char\*argv[])**

{ printf("欢迎使用简单命令解释器\n命令有mydir、mycd、mycopy、mydel、mypwd、exit\n");

while (1)

{

printf("[power by 数媒一班陈雪霞]$");

ReadCommand();

ParseCommand();

if (ExcuteCommand()==-1)

break;

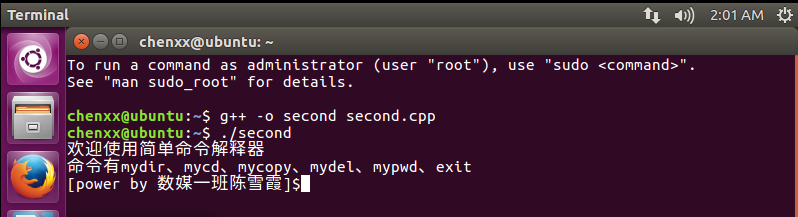
}

return 0;

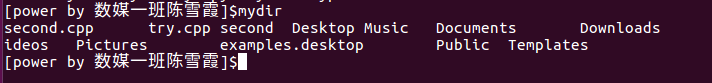
}

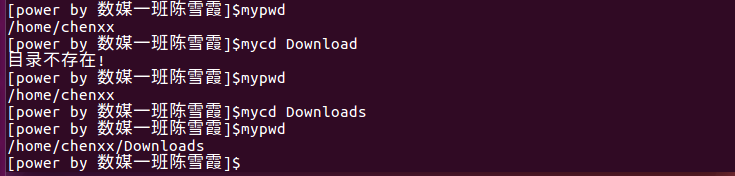
**（四）实验截图**

**1. 欢迎界面**



1. **列出目录和文件**

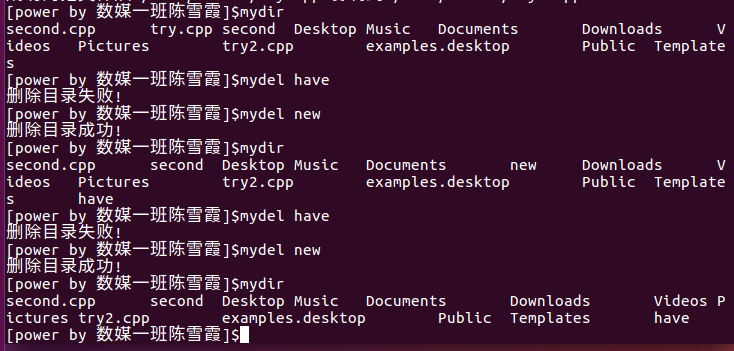
**3.改变当前目录和显示当前目录**

**4．**

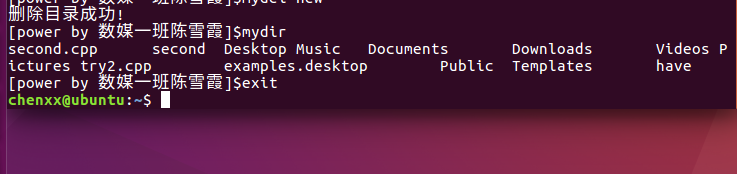
1. **在用一个文件夹内复制文件**



1. **删除目录**



1. **退出**



1. **实验小结（心得体会、遇到的问题及解决方式、未解决/需进一步研讨的问题或建议新实验方法等）**

我们这一代90后，从小接触的是windows，以前觉得Linux就跟dos一样，全是用命令窗口，相对于窗口界面来说多麻烦呀。直到学习Linux这门课以后，我才知道，原来我错了。一.为什么要学Linux呢?每个人都有不同的看法，下面我说说自己的感想吧。

首先Linux是开源的，这也是最主要的原因，想学windows，Unix，对不起我们没源代码。也正是因为这样，Linux才能够像滚雪球一样越滚越大，发展到现在这种规模。其中不乏很多IT精英的心血。我们学透以后更可以做成自己的OS!

其次，Linux简单易学，因为我们初学者只是学的基础部分，Linux的结构体系非常清晰，再加上老师循序渐进的教学以及耐心的讲解，使我们理解起来很快，短期内就基本掌握了操作和运行模式。对我们学习操作系统有很大的帮助，加深我们对OS的理解。

再次，Linux是用C语言编写的，我们有学习C语言的基础，读程序和编写代码方面存在的困难小一点，也是我们能较快掌握的原因之一。

老师一直很强调能力的培养。无论是上学期学习的计算机原理还是这学期开设的操作系统、Windows程序设计等等课程，只是课堂听老师讲解、传授知识，即使理解透彻了，也还是十分抽象，落实不到真正的技术实现上，解决不了实际问题。要真正理解掌握，真正有所体会，还是得靠自己动手，从这学期的实验也可以看出实践的重要性。

一直很难克服自己的拖延症，这次的实验也不例外。刚开始的时候虽然课程也比较紧张，但还是有一些时间的，但因为贪玩、畏难迟迟没有开始。到后来各种大实验一来，就更加觉得时间不够用了。人总是要逼一逼自己的，或许这样才能出成果。于是硬着头皮开始了shell命令解释器的实验。

一开始的时候，是先从熟悉Linux系统开始的，用惯了图形化的Windows，我甚至连终端是什么，在哪里打开都不知道。更不理解我要做的到底是个什么东西，什么是命令解释器，我最后要呈现出一个什么样的效果才算完成。然后就对照着百度到的教程，从熟悉Linux的操作和各种命令开始，到学会gcc编译，再到着手编写shell命令解释器。一步一步，走得艰难但印象深刻，缓慢而脚踏实地。