**《程序设计基础(2)》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2020** | | | **姓名** |  |
| **实验题目** | **命令行文件管理器的设计与实现** | | | | | |
| **实验时间** | **2021.4.25** | | **实验地点** | **DS1408** | | |
| **实验成绩** |  | | **实验性质** | **□验证性 □设计性 ■综合性** | | |
| 教师评价：  □算法/实验过程正确；□源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 一、实验目的  运用面向对象程序设计思想，基于Linux文件管理和C＋＋的I/O框架，实现命令行下的文件管理器。 | | | | | | |
| 二、实验项目内容  1、实现文件夹创建、删除、进入。  2、实现当前文件夹下的内容罗列。  3、实现文件拷贝和文件夹拷贝（文件夹拷贝指深度拷贝，包括所有子目录和文件）。 | | | | | | |
| 1. 实验过程或算法（源程序）   //file\_main.cpp  #include "file.cpp"  using namespace std;  int main(int argc,char \*\*argv)  {  cout<<"####################################################"<<endl;  cout<<" Welcome to the file manager"<<endl;  cout<<"####################################################"<<endl;  cout<<"\n[options]:"<<endl<<endl;  cout<<" [md] :make dictionary"<<endl;  cout<<" [rd] :remove dictionary"<<endl;  cout<<" [cd] :change dictionary "<<endl;  cout<<" [ld] :list dictionary"<<endl;  cout<<" [cp] :copy"<<endl;  cout<<" [rf] :remove file"<<endl;  cout<<" [q] :quit"<<endl;  cout<<"-----------------------------------------------------"<<endl;  cout<<"-";  string Line;  vector<string> Args;  getline(cin,Line);  Args = str\_split(Line);  file one(Args);  while(Args.size() && Args[0] != "q")  {  if(Args[0] == "md")  {  one.md() ;  }  else if(Args[0] == "rd")  {  one.rd() ;  }  else if(Args[0] == "rf")  {  one.rf() ;  }  else if(Args[0] == "cd")  {  one.cd() ;  }  else if(Args[0] == "ld")  {  one.ld() ;  }  else if(Args[0] == "cp")  {  one.cp();  }  else  {  cout<<"error: error command"<<endl;  }  cout<<"-----------------------------------------------------"<<endl;  cout<<"-";  getline(cin,Line);  Args = str\_split(Line);  one.value = Args;  }  return 0;  }  File.cpp  //file.cpp  #include "file.h"  vector<string> str\_split(string s)  {  string x;  vector<string> res;  istringstream stream(s);  stream >> x;  res.push\_back(x);  while(stream >> x){  res.push\_back(x);  }  return res;  }  vector<string> value;  void file::md(){  if(value.size()<2)  {  cout<<"error: please enter the name of dictionary"<<endl;  }  else  {  string dirName = value[1];  mode\_t mode = 0775;  if(mkdir(dirName.c\_str(),mode) == 0)  {  cout<<"make dictionary‘"<<dirName<<"’succeeded"<<endl;  }  else  {  cout<<"error: make dictionary‘"<<dirName<<"’failed"<<endl;  }  }  }  void file::rd(){  if(value.size() < 2)  {  cout<<"error: please enter the name of dictionary"<<endl;  }  else  {  string dirName = value[1];  if(rmdir(dirName.c\_str()) == 0 )  {  cout<<"remove dictionary‘"<<dirName<<"’succeeded"<<endl;  }  else  {  cout<<"error:remobve dictionary‘"<<dirName<<"’failed"<<endl;  }  }  }  void file::rf(){  if(value.size()<2)  {  cout<<"error: please enter the name of file"<<endl;  }  else  {  string fileName = value[1];  if(remove(fileName.c\_str()) == 0 )  {  cout<<"remove file‘"<<fileName<<"’succeeded"<<endl;  }  else  {  cout<<"error: remove file‘"<<fileName<<"’failed"<<endl;  }  }  }  void file::cd(){  if(value.size()<2)  {  cout<<"error: please enter the name of dictionary"<<endl;  }  else  {  string dirName = value[1];  if(chdir(dirName.c\_str())==0)  {  char curDir[300];  getcwd(curDir,300);  cout<<"changed the dictionary to"<<curDir<<endl;  }  else  {  cout<<"error: failed to change the dictioanry."<<endl;  }  }  }  void file::ld(){  DIR \* curDir;  struct dirent \*ent;  struct stat entStat;  if((curDir=opendir("."))==NULL)  {  cout<<"error: failed to open the dictionary"<<endl;  }  else  {  while((ent=readdir(curDir)) != NULL)  {  if(stat(ent->d\_name, &entStat) == 0)  {  if(entStat.st\_mode & S\_IFDIR)  {  cout<<setw(6)<<left<<"d";  cout<<right<<setw(10)<<entStat.st\_size<<left<<setw(8)<<" bytes";  cout<<ent->d\_name<<"/"<<endl;  }  else if(entStat.st\_mode & S\_IFREG)  {  cout<<setw(6)<<left<<"-";  cout<<right<<setw(10)<<entStat.st\_size<<left<<setw(8)<<" bytes";  cout<<ent->d\_name<<endl;  }  }  }  closedir(curDir);  }  }  void file::cp(){  if(value.size() < 3)  {  cout<<"error: Please enter the source file / folder name and the destination file / folder name"<<endl;  }  else  {  string sourceFileName = value[1];  string targetFileName = value[2];  ifstream sourceFile(sourceFileName.c\_str(), ios::binary);  ofstream targetFile(targetFileName.c\_str(), ios::binary);  targetFile<<sourceFile.rdbuf();  targetFile.close();  sourceFile.close();  cout<<"copy succeeded."<<endl;  }  }  File.h  //file.h  #ifndef FILE\_H  #define FILE\_H  #include <iostream>  #include <iomanip>  #include <fstream>  #include <sstream>  #include <vector>  #include <string>  #include <sys/types.h>  #include <sys/stat.h>  #include <unistd.h>  #include <dirent.h>  using namespace std;  class file  {  public:  void md();  void rd();  void rf();  void cd();  void ld();  void cp();  vector <string> value;  file(vector <string> value){  this -> value = value;  }  };  #endif | | | | | | |
| 1. 实验结果及分析和（或）源程序调试过程   截屏2021-05-16 下午11.35.01  截屏2021-05-16 下午11.35.14 | | | | | | |

备注：

1、学生应按照要求正确地撰写实验报告：

* 1. 将实验所涉及的源程序文件内容（实验操作步骤或者算法）填写在“实验过程或算法（源程序）”栏目中。
  2. 将实验所涉及源程序调试过程（输入数据和输出结果）或者实验的分析内容填写在“实验结果及分析和（或）源程序调试过程”栏目中。
  3. 在实验报告页脚的“报告创建时间：”处插入完成实验报告时的日期和时间。
  4. 学生提交实验报告时，每个实验一个电子文档，如果实验中有多个电子文档（如源程序或图形等），则压缩成一个.zip压缩包文档提交，压缩包文件名同实验报告文件名（见下条）。
  5. 提交的实验报告电子文档命名为：“年级（四位数字不要“级”字）专业（缩写：计算机科学与技术专业（计科）、信息安全专业（信息）、物联网工程（物联网））学号（八位数字）姓名＋实验序号（一位数字）。如张三（学号20165676）完成第1个Project，专业为“计算机科学与技术”专业，完成的课程设计报告命名为：**2016计科20165676张三1**，以后几次实验的报告名称以此类推。