**电 子 科 技 大 学**

**综合练习 报 告**

**学生姓名：任振华 学 号：2017060801023 指导教师：李林**

**学生E-mail：renzhenhuatime@foxmail.com**

**一、综合练习名称**

设计并实现一个基于插件框架结构的文件统计分析软件

**二、实验要求**

以插件的形式，实现若干文件统计分析功能。需要实现的功能至少包括：1.统计指定的某个文件的行数2.统计指定的某个文件的单词数3.统计指定目录下、特定后缀名的所有文件的行数。

4.每项功能使用单独的一个插件实现

其它要求：

1.被统计的文件使用ASCII编码2.使用方式：插件框架程序需要处于一个循环中，不停地接收用户发来的命令（不能使用命令行参数，指定要执行的插件功能及其参数）

**三、设计与实现**

**插件部分设计：**

我在插件系统中设计实现了6个插件(插件1，2，3满足实验要求)：

插件1：统计指定的某个文件的行数

插件2：统计指定的某个文件的单词数

插件3：统计指定目录下、特定后缀名的所有文件的行数。

插件4：统计指定目录下、特定后缀名的所有文件的单词数。

插件5：统计指定的某个文件的大小(字节数)

插件6：统计指定目录下、特定后缀名的所有文件的大小(字节数)

实现插件1：读取换行符，数量+1即可

关键代码如下：

virtual void **Func**(char \*File)*//统计文件行数*

{

    int fd;

    char temp;

    int num = 0;

    if (-1 == (fd = **open**(File, O\_RDONLY)))*//只读方式打开文件*

    {

        cout << "Can not open: " << File << endl;

        return;

    }

    while (**read**(fd, &temp, 1))

    {

        if (temp == '\n')*//每次读到换行符num++*

        {

            num++;

        }

    }

**close**(fd);*//关闭文件*

    cout << File << " line: " << num << endl;*//展示*

}

实现插件2：先写一个统计每行单词数的函数，通过把每行的单词数加起来来统计文件的单词总数

关键代码如下：

int **FuncLine**(const char \*szLine)*//统计每行单词数*

{

    int nWords = 0;

    int i = 0;

    for (; i < **strlen**(szLine); i++)

    {

        if (\*(szLine + i) != ' ')

        {

            nWords++;

            while ((\*(szLine + i) != ' ') && (\*(szLine + i) != '\0'))

            {

                i++;

            }

        }

    }

    return nWords;

}

void **Func**(char \*File)*//统计文件单词数*

{

    int nWords = 0;*//词计数变量，初始值为0*

    FILE \*fp;*//文件指针*

    char carrBuffer[1024];*//每行字符缓冲，每行最多1024个字符*

    if ((fp = **fopen**(File, "r")) == NULL)*//打开文件*

    {

        cout << "fopen error" << endl;

**exit**(-1);

    }

    while (!**feof**(fp))*//如果没有读到文件末尾*

    {

*//从文件中读一行*

        if (**fgets**(carrBuffer, sizeof(carrBuffer), fp) != NULL)

*//统计每行词数*

            nWords += **FuncLine**(carrBuffer);

    }

**fclose**(fp);*//关闭文件*

    cout << "word numbers: " << nWords << endl;

}

实现插件3：关键在于在递归查找目录下每一个文件的过程中，匹配文件后缀名是否与特定后缀名相同，匹配成功调用函数打印该文件行数

获取文件后缀名的代码：

string suffixStr = filename.substr(filename.find\_last\_of('.') + 1);//获取文件后缀

关键代码如下：

virtual void **Func**(char \*File)*//统计文件行数*

{

    int fd;

    char temp;

    int num = 0;

    if (-1 == (fd = **open**(File, O\_RDONLY)))*//只读方式打开文件*

    {

        cout << "Can not open: " << File << endl;

        return;

    }

    while (**read**(fd, &temp, 1))

    {

        if (temp == '\n')*//每次读到换行符num++*

        {

            num++;

        }

    }

**close**(fd);*//关闭文件*

    cout << File << " line: " << num << endl;*//展示*

}

virtual void **Func**(**string** path, **string** suffix)

{

    DIR \*dir;

    struct **dirent** \*ptr;

    if ((dir = **opendir**(path.**c\_str**())) == nullptr)

    {

**perror**("Open directory error...");*//打开目录文件失败*

**exit**(1);

    }

    while ((ptr = **readdir**(dir)) != nullptr)*//用readdir读取DIR dir结构体*

    {

        if (**strcmp**(ptr->d\_name, ".") == 0 || **strcmp**(ptr->d\_name, "..") == 0)*//跳过. 和 ..文件*

            continue;

        else if (ptr->d\_type == 8)

        {

            string filename = ptr->d\_name;*//转换为 string*

            string suffixStr = filename.**substr**(filename.**find\_last\_of**('.') + 1);*//获取文件后缀*

            if (suffixStr == suffix)*//匹配后缀*

            {

*//files.push\_back(path + "/" + ptr->d\_name);*

                char p[100];

**strcpy**(p, (path + "/" + ptr->d\_name).**c\_str**());

**Func**(p);*//调用函数打印该文件行数*

            }

        }

        else if (ptr->d\_type == 4)*//directory*

        {

**Func**(path + "/" + ptr->d\_name, suffix);*//递归*

        }

    }

**closedir**(dir);*//关闭流*

}

实现插件4：与实现3类似，只是把调用的函数换成打印文件单词数的函数即可。

实现插件5：统计文件大小，又因为我们假设了被统计的文件使用ASCII编码，而一个字符的ASCII码占用存储空间为1个字节，所以统计字符数即可获得文件大小。

关键代码如下：

virtual void **Func**(char \*File)

{

    int fd;

    char temp;

    int num = 0;

    if (-1 == (fd = **open**(File, O\_RDONLY)))*//只读方式打开*

    {

        cout << "Can not open: " << File << endl;

        return;

    }

    while (**read**(fd, &temp, 1))*//每读取一个字符就加1*

    {

        num++;

    }

**close**(fd);*//关闭文件*

    if (0 == num)

    {

        cout << "Empty file: " << File << endl;

    }

    cout << File << " size is : " << num << endl;*//打印文件大小*

}

实现插件6：与实现插件3，4类似，只是把调用的函数换成打印文件大小的函数而已。

**调用插件部分设计：**

设计实现了2个类，class SearchPlugin用来实现在目录plugin下搜索插件，class CPluginController用来管理插件。两个类如下：

class SearchPlugin

{

public:

SearchPlugin() {

}

~SearchPlugin() {

}

bool GetPlugin(vector<string>& PluginName);

};

class CPluginController

{

public:

CPluginController(void);

virtual ~CPluginController(void);

bool InitializeController(void);//初始化

bool UninitializeController(void);//释放

bool ProcessHelp(void);

bool ProcessRequest(int FunctionID);

bool IfProcess(char \*Function);//判断插件是否存在

bool ProcessFunction(char \*Function, char \*Document);//执行插件1，2，5

bool ProcessFunction(char \*Function, char \*Dir, char \*Suffix); //执行插件3，4，6

private:

std::vector<void \*> m\_vhForPlugin;

std::vector<IPlugin\*> m\_vpPlugin;

};

主程序部分：因为要求不使用命令行，所以使用一个while (1)来不断接收用户的命令。

执行插件的关键代码：

if (input == 'r')*//执行插件1，2，5*

{

    CPluginController ptr;

    cout << "输入文件名" << endl;

    char File[MAXSIZE];

    cin >> File;

    cout << "输入执行的插件代号" << endl;

    char Function[MAXSIZE];

    cin >> Function;

    ptr.**InitializeController**();*//初始化*

    if (ptr.**IfProcess**(Function) == false)*//判断插件是否存在*

    {

        cout << "No this plugin!" << endl;

    }

    else

    {

        ptr.**ProcessFunction**(Function, File);*//调用插件功能*

    };

    ptr.**UninitializeController**();*//释放*

    if (input == 'e')*//输入'e'时退出*

    {

**exit**(0);

        break;

    }

}

if (input == 'd')*//执行插件3，4，6*

{

    CPluginController ptr;

    cout << "输入目录名" << endl;

    char Dir[MAXSIZE];

    cin >> Dir;

    cout << "输入后缀名" << endl;

    char Suffix[MAXSIZE];

    cin >> Suffix;

    cout << "输入执行的插件代号" << endl;

    char Function[MAXSIZE];

    cin >> Function;

    ptr.**InitializeController**();*//初始化*

    if (ptr.**IfProcess**(Function) == false)*//判断插件是否存在*

    {

        cout << "No this plugin!" << endl;

    }

    else

    {

        ptr.**ProcessFunction**(Function, Dir, Suffix);*//调用插件功能*

    };

    ptr.**UninitializeController**();*//释放*

    if (input == 'e')*//输入'e'时退出*

    {

**exit**(0);

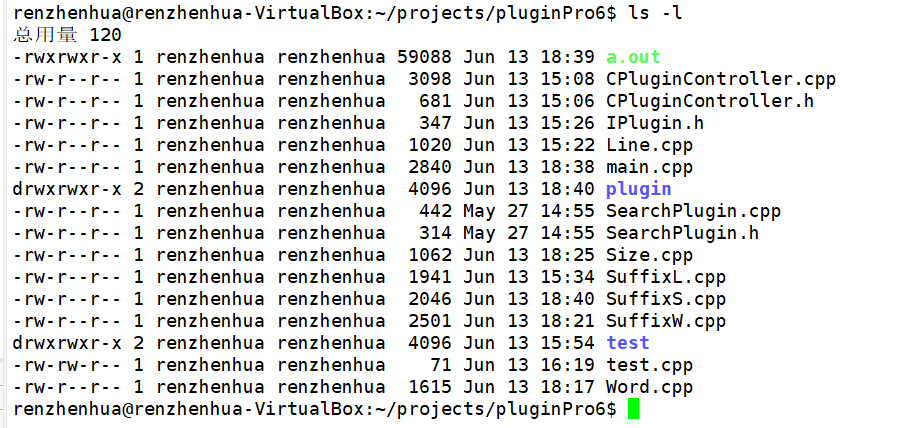
        break;

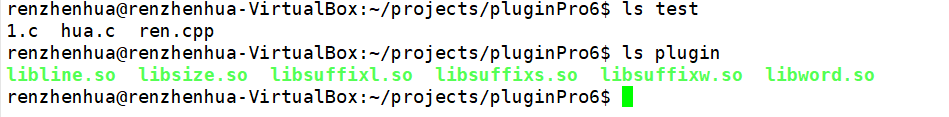
    }

}

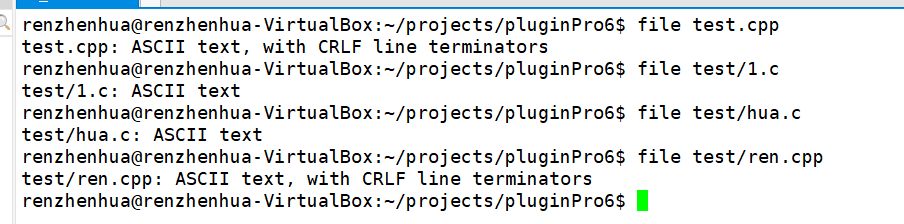
**四、测试**

项目视图如下：test为测试目录，下面有3个测试文件，test.cpp也为测试文件，plugin下面存放好了编译好的插件



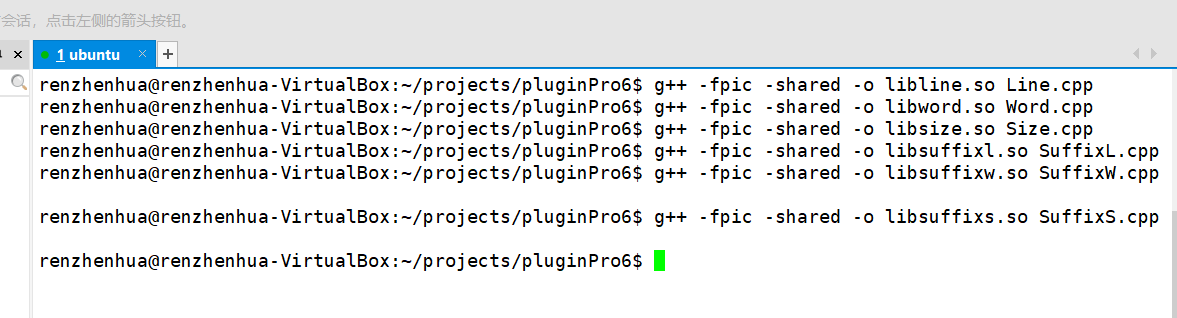


test.cpp和test目录下的文件均为ASCII text

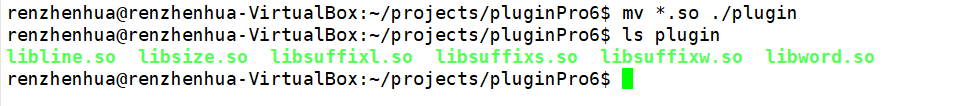


**测试流程：**

1. 编译插件，并放到plugin目录中

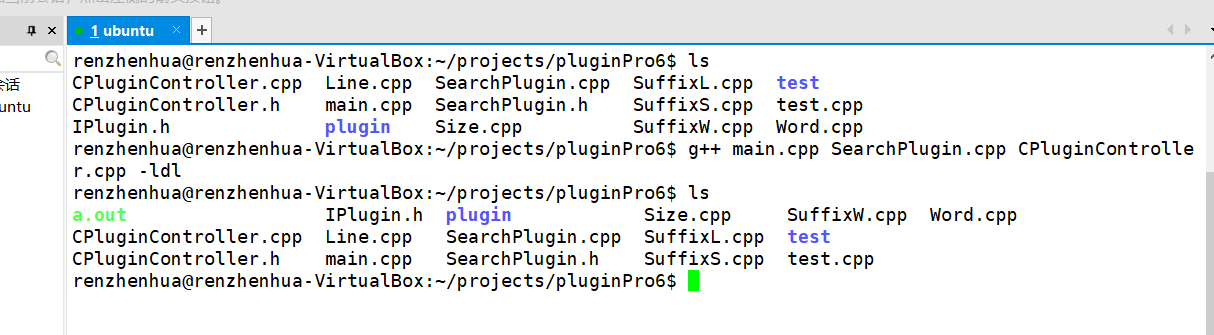


编译插件

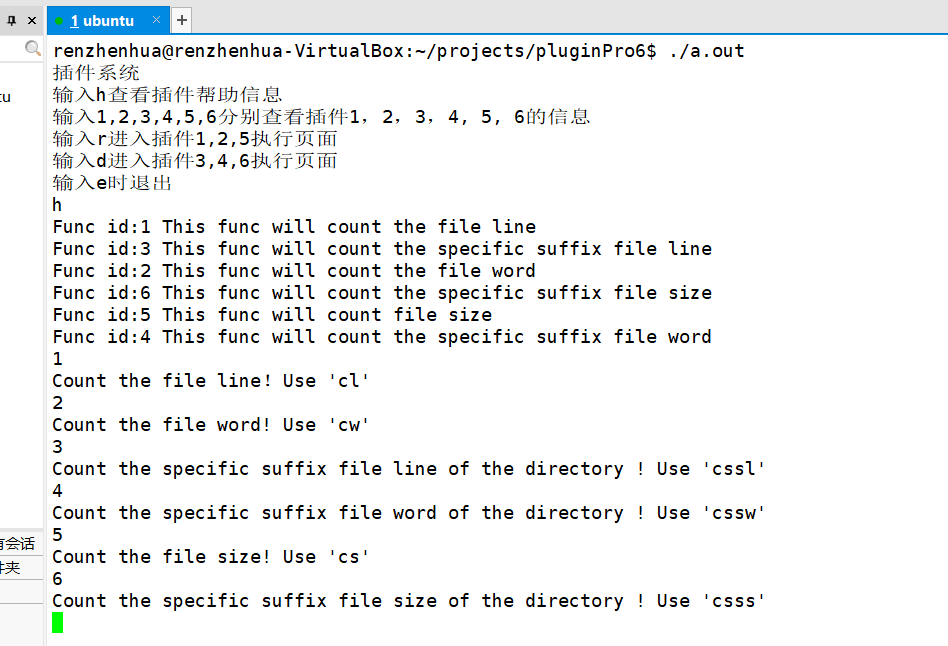


放入plugin目录中

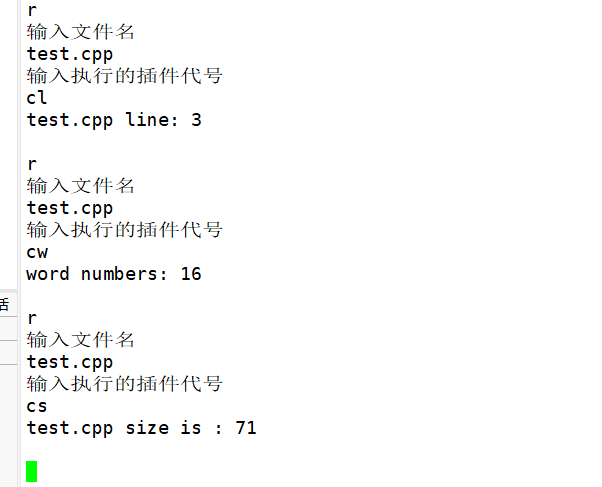
1. 编译主程序，生成a.out可执行文件



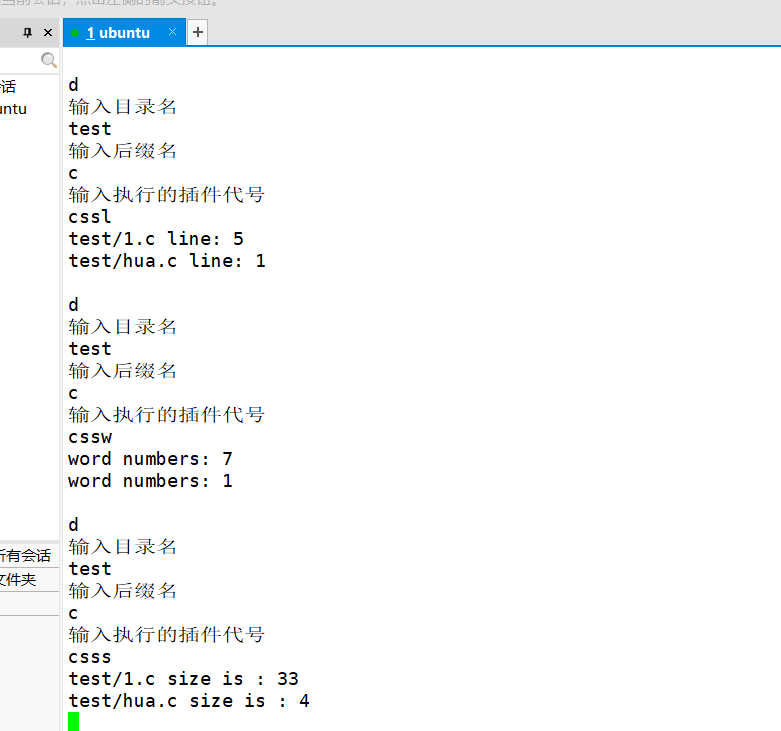
1. 执行a.out文件测试



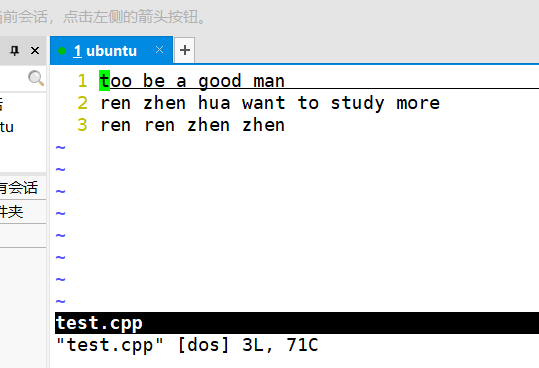
打印帮助信息

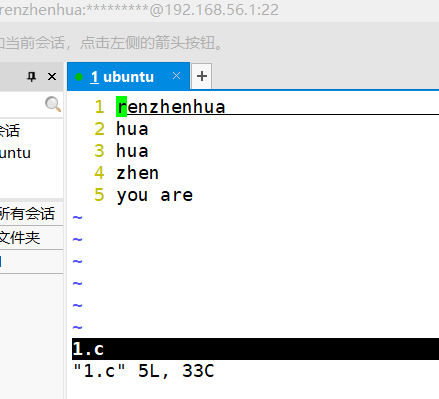


输入r执行插件1，2，5



输入d利用目录测试插件3，4，6





查看test.cpp, test/hua.c, test/1.c,验证程序正确性

**五、对本课程或本作业的建议和意见**

本次综合练习中，练习了linux下c++面向对象编程的能力，对个人水平有比较大的提高。

**六、附录**

**IPlugin.h:**

#ifndef \_IPLUGIN\_H\_

#define \_IPLUGIN\_H\_

#include<string>

class IPlugin

{

public:

virtual void Print() = 0;

virtual void Help() = 0;

virtual int GetID() = 0;

virtual char\* GetName() = 0;

virtual void Func(char\*file) = 0;

virtual void Func(std::string path, std::string suffix) = 0;

public:

IPlugin(){

}

virtual ~IPlugin(){

}

};

#endif

**Line.cpp:**

#include<iostream>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#include <string>

#include"IPlugin.h"

using namespace std;

char FUNC\_NAME[] = "cl"; //count the line

class Plugin:

public IPlugin

{

public:

Plugin()

{

}

virtual ~Plugin()

{

}

virtual void Print()

{

cout << "Count the file line！Use 'cl'" << endl;

}

virtual void Help()

{

cout <<"Func id:1 " << "This func will count the file line" <<endl;

}

virtual int GetID()

{

return 1;

}

virtual char \*GetName()

{

return FUNC\_NAME;

}

virtual void Func(char\* File)//统计文件行数

{

int fd;

char temp;

int num = 0;

if (-1 == (fd = open(File, O\_RDONLY)))//只读打开文件

{

cout << "Can not open: " << File << endl;

return;

}

while (read(fd, &temp, 1))

{

if (temp == '\n')//每次读到换行符num++

{

num++;

}

}

close(fd);//关闭文件

cout << File << " line: " << num << endl;//展示

}

virtual void Func(string path, string suffix) {

}

};

extern "C" void GetInterface(IPlugin \*\*ppPlugin)

{

static Plugin plugin;

\*ppPlugin = &plugin;

}

**Word.cpp:**

#include<iostream>

#include <unistd.h>

#include <fcntl.h>

#include <map>

#include <cstdio>

#include <string.h>

#include"IPlugin.h"

using namespace std;

char FUNC\_NAME[] = "cw"; //count the word cw

class Plugin:

public IPlugin

{

public:

Plugin()

{

}

virtual ~Plugin()

{

}

virtual void Print()

{

cout << "Count the file word! Use 'cw'" << endl;

}

virtual void Help()

{

cout <<"Func id:2 " << "This func will count the file word" <<endl;

}

virtual int GetID()

{

return 2;

}

virtual char \*GetName()

{

return FUNC\_NAME;

}

int FuncLine(const char \*szLine)

{

int nWords = 0;

int i = 0;

for (; i < strlen(szLine); i++)

{

if (\*(szLine + i) != ' ')

{

nWords++;

while ((\*(szLine + i) != ' ') && (\*(szLine + i) != '\0'))

{

i++;

}

}

}

return nWords;

}

void Func(char \*File)

{

int nWords = 0;//词计数变量，初始值为0

FILE \*fp; //文件指针

char carrBuffer[1024];//每行字符缓冲，每行最多1024个字符

if ((fp = fopen(File, "r")) == NULL)//打开文件

{

cout << "fopen error" << endl;

exit(-1);

}

while (!feof(fp))//如果没有读到文件末尾

{

//从文件中读一行

if (fgets(carrBuffer, sizeof(carrBuffer), fp) != NULL)

//统计每行词数

nWords += FuncLine(carrBuffer);

}

fclose(fp);//关闭文件

cout << "word numbers: "<<nWords << endl;

}

virtual void Func(string path, string suffix) {

}

};

extern "C" void GetInterface(IPlugin \*\*ppPlugin)

{

static Plugin plugin;

\*ppPlugin = &plugin;

}

**Size.cpp:**

#include<iostream>

#include <unistd.h>

#include <fcntl.h>

#include"IPlugin.h"

using namespace std;

char FUNC\_NAME[] = "cs"; //count the file size

class Plugin :

public IPlugin

{

public:

Plugin()

{

}

virtual ~Plugin()

{

}

virtual void Print()

{

cout << "Count the file size! Use 'cs'" << endl;

}

virtual void Help()

{

cout << "Func id:5 " << "This func will count file size" << endl;

}

virtual int GetID()

{

return 5;

}

virtual char \*GetName()

{

return FUNC\_NAME;

}

virtual void Func(char \*File)

{

int fd;

char temp;

int num = 0;

if (-1 == (fd = open(File, O\_RDONLY)))

{

cout << "Can not open: " << File << endl;

return;

}

while (read(fd, &temp, 1))

{

num++;

}

close(fd);

if (0 == num)

{

cout << "Empty file: " << File << endl;

}

cout << File << " size is : " << num << endl;

}

virtual void Func(string path, string suffix) {

}

};

extern "C" void GetInterface(IPlugin \*\*ppPlugin)

{

static Plugin plugin;

\*ppPlugin = &plugin;

}

**SuffixL.cpp:**

#include <iostream>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#include <string>

#include <vector>

#include <fstream>

#include <cstring>

#include <errno.h>

#include <cstdio>

#include <dirent.h>

#include "IPlugin.h"

using namespace std;

char FUNC\_NAME[] = "cssl"; //count the special suffix file line

class Plugin :

public IPlugin

{

public:

Plugin()

{

}

virtual ~Plugin()

{

}

virtual void Print()

{

cout << "Count the specific suffix file line of the directory ! Use 'cssl'" << endl;

}

virtual void Help()

{

cout << "Func id:3 " << "This func will count the specific suffix file line" << endl;

}

virtual int GetID()

{

return 3;

}

virtual char \*GetName()

{

return FUNC\_NAME;

}

virtual void Func(char\* File) {

int fd;

char temp;

int num = 0;

if (-1 == (fd = open(File, O\_RDONLY)))

{

cout << "Can not open: " << File << endl;

return;

}

while (read(fd, &temp, 1))

{

if (temp == '\n')

{

num++;

}

}

close(fd);

cout << File << " line: " << num << endl;

}

virtual void Func(string path, string suffix) {

DIR \*dir;

struct dirent \*ptr;

if ((dir = opendir(path.c\_str())) == nullptr) {

perror("Open directory error...");

exit(1);

}

while ((ptr = readdir(dir)) != nullptr) {

if (strcmp(ptr->d\_name, ".") == 0 || strcmp(ptr->d\_name, "..") == 0)

continue;

else if (ptr->d\_type == 8) {

string filename = ptr->d\_name;

string suffixStr = filename.substr(filename.find\_last\_of('.') + 1);//获取文件后缀

if (suffixStr == suffix) {

//files.push\_back(path + "/" + ptr->d\_name);

char p[100];

strcpy(p, (path + "/" + ptr->d\_name).c\_str());

Func(p);

}

}

else if (ptr->d\_type == 4) //directory

{

Func(path + "/" + ptr->d\_name, suffix); //递归

}

}

closedir(dir);

}

};

extern "C" void GetInterface(IPlugin \*\*ppPlugin)

{

static Plugin plugin;

\*ppPlugin = &plugin;

}

**SuffixW.cpp:**

#include <iostream>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#include <string>

#include <vector>

#include <fstream>

#include <cstring>

#include <errno.h>

#include <cstdio>

#include <dirent.h>

#include <map>

#include "IPlugin.h"

using namespace std;

char FUNC\_NAME[] = "cssw"; //count the special suffix file word

class Plugin :

public IPlugin

{

public:

Plugin()

{

}

virtual ~Plugin()

{

}

virtual void Print()

{

cout << "Count the specific suffix file word of the directory ! Use 'cssw'" << endl;

}

virtual void Help()

{

cout << "Func id:4 " << "This func will count the specific suffix file word" << endl;

}

virtual int GetID()

{

return 4;

}

virtual char \*GetName()

{

return FUNC\_NAME;

}

int FuncLine(const char \*szLine)

{

int nWords = 0;

int i = 0;

for (; i < strlen(szLine); i++)

{

if (\*(szLine + i) != ' ')

{

nWords++;

while ((\*(szLine + i) != ' ') && (\*(szLine + i) != '\0'))

{

i++;

}

}

}

return nWords;

}

void Func(char \*File)

{

int nWords = 0;//词计数变量，初始值为0

FILE \*fp; //文件指针

char carrBuffer[1024];//每行字符缓冲，每行最多1024个字符

if ((fp = fopen(File, "r")) == NULL)//打开文件

{

cout << "fopen error" << endl;

exit(-1);

}

while (!feof(fp))//如果没有读到文件末尾

{

//从文件中读一行

if (fgets(carrBuffer, sizeof(carrBuffer), fp) != NULL)

//统计每行词数

nWords += FuncLine(carrBuffer);

}

fclose(fp);//关闭文件

cout << "word numbers: " << nWords << endl;

}

virtual void Func(string path, string suffix) {

DIR \*dir;

struct dirent \*ptr;

if ((dir = opendir(path.c\_str())) == nullptr) {

perror("Open directory error...");

exit(1);

}

while ((ptr = readdir(dir)) != nullptr) {

if (strcmp(ptr->d\_name, ".") == 0 || strcmp(ptr->d\_name, "..") == 0)

continue;

else if (ptr->d\_type == 8) {

string filename = ptr->d\_name;

string suffixStr = filename.substr(filename.find\_last\_of('.') + 1);//获取文件后缀

if (suffixStr == suffix) {

//files.push\_back(path + "/" + ptr->d\_name);

char p[100];

strcpy(p, (path + "/" + ptr->d\_name).c\_str());

Func(p);

}

}

else if (ptr->d\_type == 4) //directory

{

Func(path + "/" + ptr->d\_name, suffix); //递归

}

}

closedir(dir);

}

};

extern "C" void GetInterface(IPlugin \*\*ppPlugin)

{

static Plugin plugin;

\*ppPlugin = &plugin;

}

**SuffixS.cpp:**

#include <iostream>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#include <string>

#include <vector>

#include <fstream>

#include <cstring>

#include <errno.h>

#include <cstdio>

#include <dirent.h>

#include "IPlugin.h"

using namespace std;

char FUNC\_NAME[] = "csss"; //count the special suffix file size

class Plugin :

public IPlugin

{

public:

Plugin()

{

}

virtual ~Plugin()

{

}

virtual void Print()

{

cout << "Count the specific suffix file size of the directory ! Use 'csss'" << endl;

}

virtual void Help()

{

cout << "Func id:6 " << "This func will count the specific suffix file size" << endl;

}

virtual int GetID()

{

return 6;

}

virtual char \*GetName()

{

return FUNC\_NAME;

}

virtual void Func(char \*File)

{

int fd;

char temp;

int num = 0;

if (-1 == (fd = open(File, O\_RDONLY)))

{

cout << "Can not open: " << File << endl;

return;

}

while (read(fd, &temp, 1))

{

num++;

}

close(fd);

if (0 == num)

{

cout << "Empty file: " << File << endl;

}

cout << File << " size is : " << num << endl;

}

virtual void Func(string path, string suffix) {

DIR \*dir;

struct dirent \*ptr;

if ((dir = opendir(path.c\_str())) == nullptr) {

perror("Open directory error...");

exit(1);

}

while ((ptr = readdir(dir)) != nullptr) {

if (strcmp(ptr->d\_name, ".") == 0 || strcmp(ptr->d\_name, "..") == 0)

continue;

else if (ptr->d\_type == 8) {

string filename = ptr->d\_name;

string suffixStr = filename.substr(filename.find\_last\_of('.') + 1);//获取文件后缀

if (suffixStr == suffix) {

//files.push\_back(path + "/" + ptr->d\_name);

char p[100];

strcpy(p, (path + "/" + ptr->d\_name).c\_str());

Func(p);

}

}

else if (ptr->d\_type == 4) //directory

{

Func(path + "/" + ptr->d\_name, suffix); //递归

}

}

closedir(dir);

}

};

extern "C" void GetInterface(IPlugin \*\*ppPlugin)

{

static Plugin plugin;

\*ppPlugin = &plugin;

}

**CPluginController.h:**

#ifndef \_CPLUGINCONTROLLER\_H\_

#define \_CPLUGINCONTROLLER\_H\_

#include <vector>

class IPlugin;

class CPluginController

{

public:

CPluginController(void);

virtual ~CPluginController(void);

bool InitializeController(void);//初始化

bool UninitializeController(void);//释放

bool ProcessHelp(void);

bool ProcessRequest(int FunctionID);

bool IfProcess(char \*Function);//判断插件是否存在

bool ProcessFunction(char \*Function, char \*Document);//执行插件1，2，5

bool ProcessFunction(char \*Function, char \*Dir, char \*Suffix); //执行插件3，4，6

private:

std::vector<void \*> m\_vhForPlugin;

std::vector<IPlugin\*> m\_vpPlugin;

};

#endif

**CPluginController.cpp:**

#include "CPluginController.h"

#include "SearchPlugin.h"

#include "IPlugin.h"

#include "dlfcn.h"

CPluginController::CPluginController(void)

{

}

CPluginController::~CPluginController(void)

{

}

bool CPluginController::InitializeController(void)

{

//存放所有插件的文件名

std::vector<std::string> PluginName;

//定义插件搜索类对象

SearchPlugin ptr;

//获取所有的插件文件名

if (!ptr.GetPlugin(PluginName))

return false;

for (unsigned int i = 0; i < PluginName.size(); i++)

{

typedef int(\*PLUGIN\_CREATE)(IPlugin\*\*);

PLUGIN\_CREATE GetIFC;

IPlugin \*pPlugin = NULL;

//打开动态链接库文件

void\* handle = dlopen(PluginName[i].c\_str(), RTLD\_LAZY);

if (handle != NULL)

{

m\_vhForPlugin.push\_back(handle);

//获取导出的接口对象指针

GetIFC = (PLUGIN\_CREATE)dlsym(handle, "GetInterface");

if (NULL != GetIFC)

{

(GetIFC)(&pPlugin);

if (pPlugin != NULL)

{

m\_vpPlugin.push\_back(pPlugin);

}

}

}

}

return true;

}

bool CPluginController::UninitializeController()

{

for (unsigned int i = 0; i < m\_vhForPlugin.size(); i++)

{

dlclose(m\_vhForPlugin[i]);

}

return true;

}

bool CPluginController::ProcessRequest(int FunctionID)

{

for (unsigned int i = 0; i < m\_vpPlugin.size(); i++)

{

if (m\_vpPlugin[i]->GetID() == FunctionID)

{

m\_vpPlugin[i]->Print();

break;

}

}

return true;

}

bool CPluginController::ProcessHelp(void)

{

//存放所有插件的文件名

std::vector<std::string> PluginName;

//定义插件搜索类对象

SearchPlugin ptr;

//获取所有的插件文件名

if (!ptr.GetPlugin(PluginName))

return false;

for (unsigned int i = 0; i < PluginName.size(); i++)

{

typedef int(\*PLUGIN\_CREATE)(IPlugin\*\*);

PLUGIN\_CREATE GetIFC;

IPlugin \*pPlugin = NULL;

//打开动态链接库文件

void\* handle = dlopen(PluginName[i].c\_str(), RTLD\_LAZY);

if (handle != NULL)

{

//获取导出的接口对象指针

GetIFC = (PLUGIN\_CREATE)dlsym(handle, "GetInterface");

if (NULL != GetIFC)

{

(GetIFC)(&pPlugin);

if (pPlugin != NULL)

{

pPlugin->Help();

}

}

dlclose(handle);

}

}

return true;

}

bool CPluginController::IfProcess(char \*Function)//判断插件是否存在

{

unsigned int i;

for (i = 0; i < m\_vpPlugin.size(); i++)

{

if (strcmp(Function, m\_vpPlugin[i]->GetName()) == 0)

{

break;

}

};

if (i < m\_vpPlugin.size())//插件存在

{

return true;

}

else

{

return false;

}

}

bool CPluginController::ProcessFunction(char \*Function, char\*File)//执行插件1，2，5功能

{

for (unsigned int i = 0; i < m\_vpPlugin.size(); i++)

{

if (strcmp(Function, m\_vpPlugin[i]->GetName()) == 0)

{

m\_vpPlugin[i]->Func(File);//插件功能

break;

}

}

return true;

}

bool CPluginController::ProcessFunction(char \*Function, char \*Dir, char \*Suffix)//执行插件3，4，6的功能

{

for (unsigned int i = 0; i < m\_vpPlugin.size(); i++)

{

if (strcmp(Function, m\_vpPlugin[i]->GetName()) == 0)

{

m\_vpPlugin[i]->Func(Dir, Suffix);//插件功能

break;

}

}

return true;

}

**SearchPlugin.h:**

#ifndef \_SEARCHPLUGIN\_H\_

#define \_SEARCHPLUGIN\_H\_

#include<iostream>

#include<vector>

#include<string>

#include<dirent.h>

#include<cstring>

using namespace std;

class SearchPlugin

{

public:

SearchPlugin() {

}

~SearchPlugin() {

}

bool GetPlugin(vector<string>& PluginName);

};

#endif // !\_SEARCHPLUGIN\_H\_

**SearchPlugin.cpp:**

#include "SearchPlugin.h"

bool SearchPlugin::GetPlugin(vector<string>& PluginName) {

DIR\*dir = opendir("./plugin");

if (0 == dir) {

return false;

}

while (1) {

struct dirent \*ptr = readdir(dir);

if (0 == ptr) {

break;

}

if (strcmp(ptr->d\_name, ".") == 0 || strcmp(ptr->d\_name, "..") == 0)

continue;

string str = "./plugin/";

str = str + ptr->d\_name;

PluginName.push\_back(str);

}

closedir(dir);

return true;

}

**main.cpp:**

#include <iostream>

#include <cstdio>

#include <string.h>

#include <stdlib.h>

#include "CPluginController.h"

#define MAXSIZE 100

using namespace std;

int main()

{

char input;

cout << "插件系统" << endl;

cout << "输入h查看插件帮助信息" << endl;

cout << "输入1,2,3,4,5,6分别查看插件1，2，3，4, 5, 6的信息" << endl;//1,2,3为所需要实现的插件功能

cout << "输入r进入插件1,2,5执行页面" << endl; //4,5,6为额外增加的插件功能

cout << "输入d进入插件3,4,6执行页面" << endl;

cout << "输入e时退出" << endl;

while (1) {

scanf("%c", &input);

getchar();

if (input == 'h') {

CPluginController pc;

pc.ProcessHelp();

}

if (input == '1') {

CPluginController pc;

pc.InitializeController();

pc.ProcessRequest(1);

pc.UninitializeController();

}

if (input == '2') {

CPluginController pc;

pc.InitializeController();

pc.ProcessRequest(2);

pc.UninitializeController();

}

if (input == '3') {

CPluginController pc;

pc.InitializeController();

pc.ProcessRequest(3);

pc.UninitializeController();

}

if (input == '4') {

CPluginController pc;

pc.InitializeController();

pc.ProcessRequest(4);

pc.UninitializeController();

}

if (input == '5') {

CPluginController pc;

pc.InitializeController();

pc.ProcessRequest(5);

pc.UninitializeController();

}

if (input == '6') {

CPluginController pc;

pc.InitializeController();

pc.ProcessRequest(6);

pc.UninitializeController();

}

if (input == 'r') {

CPluginController ptr;

cout << "输入文件名" << endl;

char File[MAXSIZE];

cin >> File;

cout << "输入执行的插件代号" << endl;

char Function[MAXSIZE];

cin >> Function;

ptr.InitializeController();

if (ptr.IfProcess(Function) == false)//判断插件是否存在

{

cout << "No this plugin!" << endl;

}

else

{

ptr.ProcessFunction(Function, File);

};

ptr.UninitializeController();

if (input == 'e') {

exit(0);

break;

}

}

if (input == 'd') {

CPluginController ptr;

cout << "输入目录名" << endl;

char Dir[MAXSIZE];

cin >> Dir;

cout << "输入后缀名" << endl;

char Suffix[MAXSIZE];

cin >> Suffix;

cout << "输入执行的插件代号" << endl;

char Function[MAXSIZE];

cin >> Function;

ptr.InitializeController();

if (ptr.IfProcess(Function) == false)//判断插件是否存在

{

cout << "No this plugin!" << endl;

}

else

{

ptr.ProcessFunction(Function, Dir, Suffix);

};

ptr.UninitializeController();

if (input == 'e') {

exit(0);

break;

}

}

if (input == 'e') {

exit(0);

break;

}

}

}