C++ 跨平台文件、目录操作

介绍

常见问题

1. 统计项目文件数量、统计项目代码行数
2. 创建目录、创建多级目录，删除目录、复制目录、移动目录等
3. 读取文件、复制文件、移动文件、删除文件、获取文件所在目录等

常用功能封装

1. 基于 C++11 的技术
2. 常用文件操作
3. 常用目录操作
4. 常用文件流读写操作
5. 跨平台

为什么要自己封装呢?

1. 标准库 fstream 功能强大，但是使用方面不是很方便
2. 目录的操作需要 C++17 的编译器支持，但很多时候生产环境的编译器版本没有那么高

文件操作

类：File

成员函数	描述
name	获取文件名
dir	获取文件所在目录

create	创建文件
remove	删除文件
copy	复制文件
rename	文件重命名(移动文件)
exists	判断文件是否存在
clear	清空文件内容
line	获取文件行数
size	获取文件大小
read	一次性读取文件全部内容
write	一次性写入文件

获取文件名

函数：name

```cpp
#include <iostream>
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    std::cout << file.name() << std::endl;
    return 0;
}
```

获取文件所在目录

函数：dir

```cpp
#include <iostream>
#include <fs/file.h>
```

```cpp
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    std::cout << file.dir() << std::endl;
    return 0;
}
```

创建一个空文件

函数：create

```cpp
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    file.create();
    return 0;
}
```

删除文件

函数：remove

```cpp
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    file.remove();
    return 0;
}
```

复制文件

函数：copy

```cpp
#include <iostream>
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    if (file.copy("./../test/1.txt"))
    {
        std::cout << "copy file ok" << std::endl;
    }
    else
    {
        std::cout << "copy file error" << std::endl;
    }
    return 0;
}
```

文件重命名(移动文件)

函数：rename

```cpp
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    file.rename("./../test/a/b/c/2.txt");
    return 0;
}
```

判断文件是否存在

函数：exists

```cpp
#include <iostream>
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    if (file.exists())
    {
        std::cout << "file exists" << std::endl;
    }
    else
    {
        std::cout << "file not exists" << std::endl;
    }
    return 0;
}
```

清空文件内容

```cpp
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    file.clear();
    return 0;
}
```

获取文件行数

函数：line

```cpp
#include <iostream>
#include <fs/file.h>
using namespace yazi::fs;

int main()
```

```
{
    File file("./../test/a/b/c/1.txt");
    std::cout << file.line() << std::endl;
    return 0;
}
```

获取文件大小

函数：size

```
#include <iostream>
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    std::cout << file.size() << std::endl;
    return 0;
}
```

读取文件全部内容

函数：read

```
#include <iostream>
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    std::cout << file.read() << std::endl;
    return 0;
}
```

写入文件

函数：write

```
#include <iostream>
#include <fs/file.h>
using namespace yazi::fs;

int main()
{
    File file("./../test/a/b/c/1.txt");
    file.write("hello world");
    return 0;
}
```

目录操作

类：Directory

成员函数	描述
path	获取目录的路径
create	创建一个空目录(含子目录)
remove	删除目录(含子目录)
copy	复制目录(含子目录)
rename	目录重命名(移动目录)
exists	判断目录是否存在
clear	清空目录(含子目录)
file	获取目录(含子目录)下全部文件
count	获取目录(含子目录)下包含多少个文件
line	获取目录(含子目录)下全部文件的行数
size	获取目录(含子目录)下全部文件的大小

获取目录的路径

函数：path

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/a/b/c/");
    std::cout << dir.path() << std::endl;
    return 0;
}
```

创建一个空目录(含子目录)

函数：create

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/a/b/c/");
    std::cout << dir.create() << std::endl;
    return 0;
}
```

删除目录(含子目录)

函数：remove

```cpp
#include <iostream>
```

```cpp
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/a/b/c/");
    std::cout << dir.remove() << std::endl;
    return 0;
}
```

复制目录(含子目录)

函数：copy

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/a/b/c/");
    dir.copy("./../test/");
    return 0;
}
```

目录重命名(移动目录)

函数：rename

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/a/");
    dir.rename("./../test/x/");
    return 0;
}
```

判断目录是否存在

函数：exists

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/");
    if (dir.exists())
    {
        std::cout << "dir exists" << std::endl;
    }
    else
    {
        std::cout << "dir not exists" << std::endl;
    }
    return 0;
}
```

清空目录(含子目录)

函数：clear

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/");
    dir.clear();
    return 0;
}
```

获取目录(含子目录)下全部文件

函数：file

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/");
    auto files = dir.file();
    for (const auto & file : files)
    {
        std::cout << file.name() << std::endl;
    }
    return 0;
}
```

获取目录(含子目录)下包含多少个文件

函数：count

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/");
    std::cout << dir.count() << std::endl;
    return 0;
}
```

获取目录(含子目录)下全部文件的行数

函数：line

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/");
    std::cout << dir.line() << std::endl;
    return 0;
}
```

获取目录(含子目录)下全部文件的大小

函数：size

```cpp
#include <iostream>
#include <fs/directory.h>
using namespace yazi::fs;

int main()
{
    Directory dir("./../test/");
    std::cout << dir.size() << std::endl;
    return 0;
}
```

文件流读取操作

类：FileReadStream

成员函数	描述
open	打开文件流
close	关闭文件流
read_char	逐个字符读取文件

read_line	逐行读取文件

逐个字符读取文件

函数：read_char

```
#include <iostream>
#include <fs/file_read_stream.h>
using namespace yazi::fs;

int main()
{
    FileReadStream frs("./../test/1.txt");
    if (!frs.open())
    {
        std::cout << "file open error" << std::endl;
        return -1;
    }
    char data;
    while (frs.read_char(data))
    {
        std::cout << data;
    }
    return 0;
}
```

逐行读取文件

函数：read_line

```
#include <iostream>
#include <fs/file_read_stream.h>
using namespace yazi::fs;

int main()
{
    FileReadStream frs("./../test/1.txt");
    if (!frs.open())
```

```
    {
        std::cout << "file open error" << std::endl;
        return -1;
    }
    std::string data;
    while (frs.read_line(data))
    {
        std::cout << data << std::endl;
    }
    return 0;
}
```

文件流写入操作

类：FileWriteStream

成员函数	描述
open	打开文件流
close	关闭文件流
write_char	逐个字符写入文件
write_line	逐行写入文件

逐个字符写入文件

函数：write_char

```
#include <iostream>
#include <fs/file_write_stream.h>
using namespace yazi::fs;

int main()
{
    FileWriteStream fws("./../test/2.txt");
    if (!fws.open())
```

```
    {
        std::cout << "file open error" << std::endl;
        return -1;
    }
    fws.write_char('a');
    fws.write_char('b');
    fws.write_char('c');
    return 0;
}
```

逐行写入文件

函数：write_line

```
#include <iostream>
#include <fs/file_write_stream.h>
using namespace yazi::fs;

int main()
{
    FileWriteStream fws("./../test/3.txt");
    if (!fws.open())
    {
        std::cout << "file open error" << std::endl;
        return -1;
    }
    fws.write_line("abc");
    fws.write_line("123");
    fws.write_line("456");
    return 0;
}
```

跨平台

文件操作

#include <fstream>

C++11 标准库、跨平台

目录操作

#include <dirent.h>

Linux 平台支持，Windows平台不支持

解决方案：

1、C++17 std::filesystem

2、https://github.com/tronkko/dirent

考虑到客观因素，我目前采取的是第 2 种方案。

完美！！！