·四川大学计算机学院学院

实 验 报 告

学号：2022141460176 姓名：杨一舟 专业：计算机科学与技术 第 13 周

课程名称	操作系统课程设计	实验课时	2
实验项目	Linux 文件系统管理（续）	实验时间	2024.5.27
实验目的	1、熟悉 Linux 文件管理的常见命令 2、理解 Linux 文件管理机制 3、熟悉目录的创建、维护和文件操作		
实验环境	Ubuntu 操作系统		
实验内容	**实验内容** 1.补全 loadDirectoryFromDisk() 函数 2.补全 writeData2File() 3.补全 saveDirectory2Disk()函数 测试： 创建一个文件，然后以 ASCII 码形式写入姓名拼音和学号。 读取该文件，输出文件 Inode 信息和文件内容。		

实验步骤

一、补全代码

1. loadDirectoryFromDisk 负责从磁盘上的指定目录读取文件和子目录信息，并将这些信息整理存储。

```
SDirectory loadDirectoryFromDisk(const char* vDirectoryName)
{
    SDirectory DirStructure;

    int inodeNum = findFileInodeNum(vDirectoryName);

    SInode DirInode = loadInodeFromDisk(inodeNum);

    for(int i = 0; i < g_NumDirectBlocks; ++i)
    {
        char* blockPtr = g_Disk + DirInode.DirectBlock[i] * g_BlockSize;

        for(int j = 0; j < g_BlockSize / sizeof(SFileEntry); ++j)
        {
            SFileEntry entry;
            memcpy(&entry, blockPtr + j*sizeof(SFileEntry), sizeof(SFileEntry));
            if(entry.IsInUse)
            {
                strncpy(DirStructure.FileSet[DirStructure.FileCount].FileName,
entry.FileName, g_MaxFileNameLen);
                DirStructure.FileSet[DirStructure.FileCount].InodeNum = entry.InodeNum;
                DirStructure.FileSet[DirStructure.FileCount].IsInUse = true;
                ++DirStructure.FileCount;
            }
        }
    }

    return DirStructure;
}
```

2. writeData2File 把一段数据直接写入到硬盘上的文件中。

```
bool writeDataToInode(int vInodeNum, const char* data, size_t dataSize)
{
    SInode inode = loadInodeFromDisk(vInodeNum);
    if(inode.Size + dataSize > g_MaxFileSize)
    {
        std::cerr << "文件大小超过最大限制。\n";
        return false;
    }

    inode.Size += dataSize;

    saveInode2Disk(inode, vInodeNum);
    return true;
}
```

3. saveDirectory2Disk 将内存中维护的目录结构转换成磁盘上可读的文本文件。

```
void saveDirectoryToDisk(const SDirectory& vDirectory, int vInodeNum)
{
    SInode inode = loadInodeFromDisk(vInodeNum);
    inode.Size = vDirectory.FileCount * sizeof(SFileEntry);
    saveInode2Disk(inode, vInodeNum);
}
```

二、测试验证

1. 编译源代码

在 Linux 环境下使用 make 命令（Makefile 已经配置好）来编译代码：

根据现有的 Makefile 配置更新后的各个文件，上述命令将编译 BitMap.cpp、Directory.cpp、FileSystem.cpp 和 main.cpp，生成可执行文件 FileSystem.out。

```
mountain@Lumous:~/OS_projects/文件创建和删除$ make
g++ -c BitMap.cpp
g++ -c Directory.cpp
g++ -c FileSystem.cpp
g++ -c main.cpp
g++ -o FileSystem.out BitMap.o Directory.o FileSystem.o main.o
```

<table>
<tr><td></td><td>2. 执行可执行文件
在终端中，运行编译后的可执行文件：
./FileSystem.out
  </td></tr>
<tr><td>实验结果</td><td> 
运行结果如图所示，成功创建并将数据写入了 test 文件，读取到了正确的内容与一些与 Innode 相关的信息，符合预期结果</td></tr>
<tr><td>小　结</td><td>本次实验围绕操作系统中文件及目录管理的核心功能展开，通过自定义数据结构及模拟磁盘操作，实现了从磁盘加载目录、写入文件数据及保存目录结构到磁盘的关键过程，体现了数据序列化的重要性。实验不仅加深了我对文件系统底层操作的理解，也锻炼了直接操作内存映射磁盘数据的能力，为深入学习操作系统原理奠定了实践基础。</td></tr>
<tr><td>指导老师评　议</td><td>成绩评定：　　指导教师签名：</td></tr>
</table>