

四川大学计算机学院、软件学院

实验报告

学号：2023141460321 姓名：孙谦昊 专业：计算机科学与技术 班级：行政七班 第 13 周

课程名称	操作系统实验	实验课时	8-9 节
实验项目	文件系统	实验时间	2025.05.20
实验目的	1.熟悉 Linux 文件系统管理 2.理解位视图、Inode、文件创建与删除等概念		
实验环境	VM WorkStation Pro、ubuntu-24.04.2-desktop-amd64		
实验内容（算法、程序、步骤和方法）	<p>1.补充 BitMap 中未实现的函数：</p> <p>将 BitMap 中的 isAvailableBitAt(int vBitPosition, const SBitMap& vBitMap) 函数补充为：</p> <pre>bool isAvailableBitAt(int vBitPosition, const SBitMap& vBitMap) { return ! (vBitMap.pMapData[vBitPosition / g_NumBitsInWord] & (1 << (vBitPosition % g_NumBitsInWord))); }</pre> <p>将 BitMap 中的 clearBitAt(int vBitPosition, SBitMap& vBitMap) 函数补充为：</p> <pre>void clearBitAt(int vBitPosition, SBitMap& vBitMap) { vBitMap.pMapData[vBitPosition / g_NumBitsInWord] &= ~(1 << (vBitPosition % g_NumBitsInWord)); }</pre> <p>将 BitMap 中的 findAndSetAvailableBit(SBitMap& vBitMap) 函数补充为：</p> <pre>int findAndSetAvailableBit(SBitMap& vBitMap) {</pre>		

```

// printfBitMap(voBitMap);
for (int i = 0; i < voBitMap.NumBits ; i++) {
// printf("%d\n", voBitMap.pMapData[i / g_NumBitsInWord]);
// printf("%d\n", 1 << (i % g_NumBitsInWord));
if ((voBitMap.pMapData[i / g_NumBitsInWord] & (1 <<
(i % g_NumBitsInWord))) == 0)
{
    markBitAt(i, voBitMap);
    return i;
}
return -1;
}

```

将 **BitMap** 中的 `countClearBits(const SBitMap& vBitMap)` 函数补充为:

```

int countClearBits(const SBitMap& vBitMap)
{
    int res = 0;
    for (int i = 0; i < vBitMap.NumBits; i++)
    {
        if ((vBitMap.pMapData[i / g_NumBitsInWord] & (1
        << (i % g_NumBitsInWord) ) ) == 0) res++;
    }
    return res;
}

```

2. 补充通过 **Inode** 回收磁盘块的函数;

将 **Inode** 中的 `deallocateDisk(const SInode& vInode, SBitMap & vIOCBitMap)` 函数补充为:

```

void deallocateDisk(const SInode& vInode, SBitMap& vIOCBitMap)
{
    for (int i = 0; i < vInode.NumBlocks; ++i)
    {
        clearBitAt(vInode.BlockNums[i], vIOCBitMap);
    }
}

```

3.补充 Directory.cpp 中的函数;

将 Directory 中的 findFileIndex(const char* vFileName, const SDirectory& vDirectory) 函数补充为:

```
int findFileIndex(const char* vFileName, const SDirectory
& vDirectory)
{
    for (int i = 0;i < g_MaxNumFiles ;++i)
    {
        if (strcmp(vDirectory. FileSet[i].FileName, vFileN
ame) == 0)
        {
            return i;
        }
    }
    return -1;
}
```

将 Directory 中的 addFile2Directory(const char* vFileName, short vInodeNum, SDirectory& voDirectory) 函数补充为:

```
bool addFile2Directory(const char* vFileName, short vInod
eNum, SDirectory& voDirectory)
{
    for (int i = 0;i < g_MaxNumFiles ;++i)
    {
        if (!voDirectory. FileSet[i].IsInUse)
        {
            voDirectory. FileSet[i]. IsInUse = true;
            memcpy(voDirectory. FileSet[i].FileName, vFi
lename, g_MaxFileNameLen);
            voDirectory. FileSet[i].InodeNum = vInodeNu
m;
            return true;
        }
    }
    return false;
}
```

4.补充删除文件的函数（本次实验报告实现了删除目录与 Inode 的 link 数大于 1 的情况）

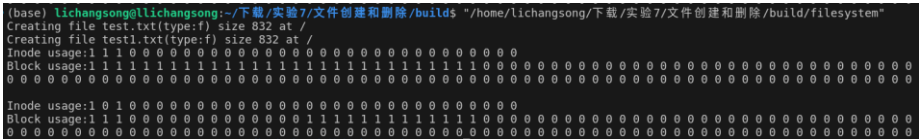
将相应函数补充为：

```
bool removeFile(const char* vFileName, int vDirInodeNum)
{
    SDirectory TempDirectory = loadDirectoryFromDisk(vDirInodeNum);
    int index = findFileIndex(vFileName, TempDirectory);
    if (index == -1) return false;
    int tempInodeNum = TempDirectory.FileSet[index].InodeNum;
    SInode tempInode = loadInodeFromDisk(tempInodeNum);
    if (tempInode.FileType == 'd')
    {
        SDirectory temp = loadDirectoryFromDisk(tempInodeNum);
    }
    for (int i = 0; i < g_MaxNumFiles; i++)
    {
        removeFile(temp.FileSet[i].FileName, tempInodeNum);
    }
    if (tempInode.NumLinks == 1)
    {
        SBitMap InodeBitMap;
        createEmptyBitMap(InodeBitMap, g_NumInodes);
        memcpy(InodeBitMap.pMapData, g_Disk+g_BlockBitMapSize, g_InodeBitMapSize);

        SBitMap DataBlockBitMap;
        createEmptyBitMap(DataBlockBitMap, g_NumBlocks);
        memcpy(DataBlockBitMap.pMapData, g_Disk, g_BlockBitMapSize);

        if (isAvailableBitAt(tempInodeNum, InodeBitMap)) return false;

        deallocateDisk(tempInode, DataBlockBitMap);
        clearBitAt(tempInodeNum, InodeBitMap);
    }
}
```

	<pre>memcpy(g_Disk, DataBlockBitMap.pMapData, g_BlockBitMapSize); memcpy(g_Disk+g_BlockBitMapSize, InodeBitMap.pMapData, g_InodeBitMapSize); } else { tempInode. NumLinks -= 1; } removeFileFromDirectory(vFileName, TempDirectory); saveDirectory2Disk(vDirInodeNum, TempDirectory); return true; }</pre>
结 论 (结 果)	<p>运行 main.cpp 的结果如下</p>  <p>1. 创建文件后 Inode 除根目录外使用了两个，符合预期，且正确处理了重复文件的创建。</p> <p>2. 删除文件后 Inode 除根目录外使用了后一个，符合预期，且正确处理了重复文件的删除。</p>
小 结	实验通过相关任务的完成熟悉 Linux 文件系统管理，并且深入理解位视图、Inode、文件创建与删除等概念
指导老师 老师评 议	<div>成绩评定:</div> <div>指导教师签名:</div>