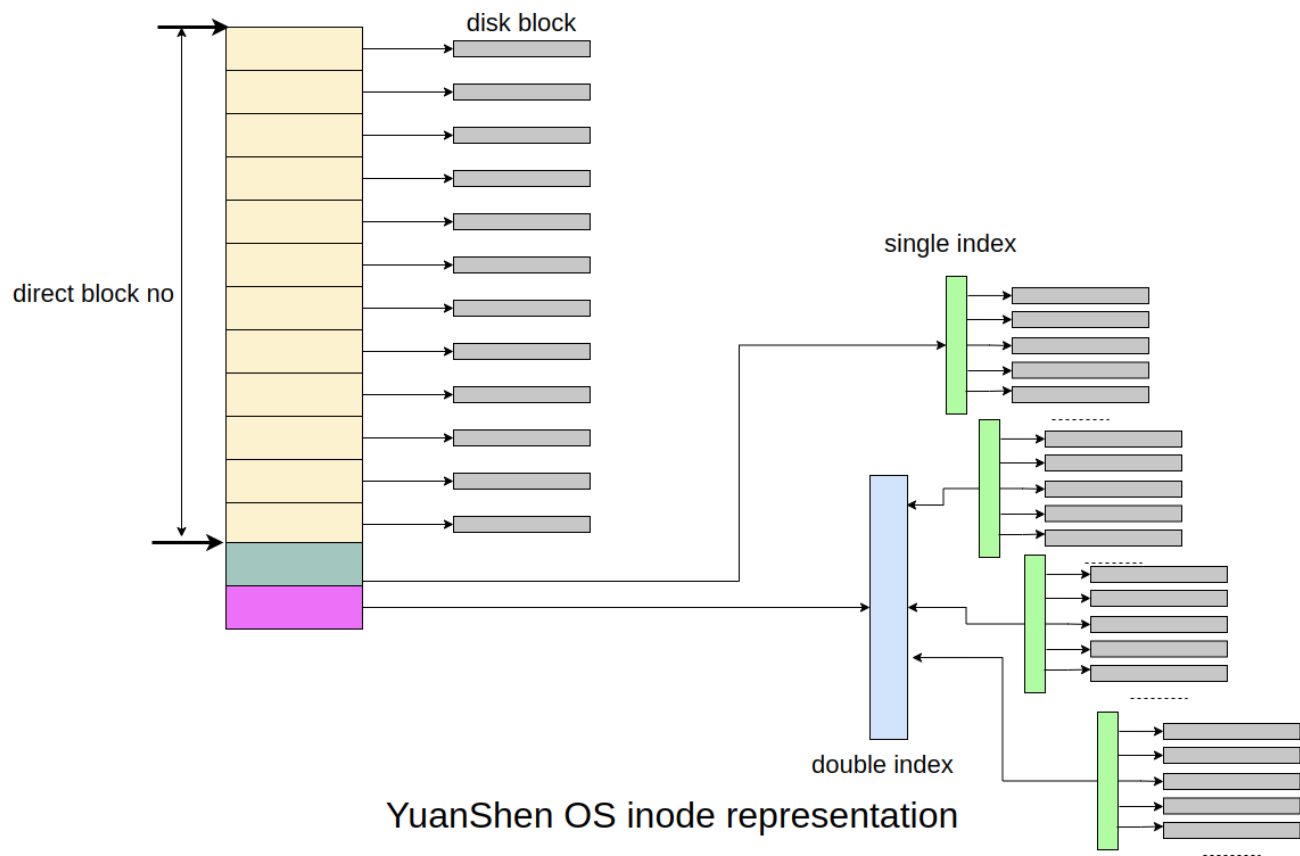


文件系统

注: 本文绘制的所有图片由组员绘制。

YuanShen OS在文件系统方向的改进参照了MIT操作系统的文件系统实验，为原生的xv6文件系统增加了二级索引页与符号链接的支持。



1 二级索引页

xv6的inode仅能维护至多12个直接磁盘块号以及1个一级索引块，这限制了xv6最大的文件大小为268个磁盘块，这显然是不够使用的。dinode结构体代码如下所示：

```
// kernel/fs.c
// On-disk inode structure
struct dinode {
    short type;           // File type
    short major;          // Major device number (T_DEVICE only)
    short minor;          // Minor device number (T_DEVICE only)
    short nlink;          // Number of links to inode in file system
    uint size;             // Size of file (bytes)
    uint addrs[NDIRECT+1]; // Data block addresses
};
```

YuanShen OS在xv6的基础上增加了一个二级索引块，让我们最大能创建65803块磁盘块大小的文件。为此我们修改了inode的格式,让inode能多保存一个磁盘块号,代码如下所示:

```
struct dinode {
    short type;           // File type
    short major;          // Major device number (T_DEVICE only)
    short minor;          // Minor device number (T_DEVICE only)
    short nlink;          // Number of links to inode in file system
    uint size;            // Size of file (bytes)
    uint addrs[NDIRECT+2]; // Data block addresses
};
```

为此我们修改了bmap函数，它的功能是获取inode中第n个块的块号。代码如下所示：

```
//fs.c:bmap 新增代码
if (bn < NDBL_INDIRECT) {
    if ((addr = ip->addrs[NDIRECT + 1]) == 0) {
        addr = balloc(ip->dev);
        if (addr == 0)
            return 0;
        ip->addrs[NDIRECT + 1] = addr;
    }

    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;

    uint index1 = bn / NINDIRECT;

    if ((addr = a[index1]) == 0) {
        addr = balloc(ip->dev);
        if (addr == 0)
            return 0;
        a[bn / NINDIRECT] = addr;
        log_write(bp);
    }
    brelse(bp);

    bp = bread(ip->dev, addr);
    a = (uint *)bp->data;

    uint index2 = bn % NINDIRECT;

    if ((addr = a[index2]) == 0) {
        addr = balloc(ip->dev);
        if (addr == 0)
            return 0;
        a[bn % NINDIRECT] = addr;
        log_write(bp); // Record changes in the log
    }
    brelse(bp);
```

```
    return addr; // Returns the actual data block
}
```

我们还修改了itrunc，让其能够识别二级索引，正确释放inode关联的所有块号。代码如下所示：

```
// fs.c:itrunc 新增释放二级索引块
if (ip->addr[NDIRECT + 1]) {
    bp = bread(ip->dev, ip->addr[NDIRECT + 1]);
    a = (uint*)bp->data;
    for (i = 0; i < NINDIRECT; ++i) {
        if (a[i] == 0) continue;
        struct buf* bp2 = bread(ip->dev, a[i]);
        uint* b = (uint*)bp2->data;
        for (j = 0; j < NINDIRECT; ++j) {
            if (b[j])
                bfree(ip->dev, b[j]);
        }
        brelse(bp2);
        bfree(ip->dev, a[i]);
        a[i] = 0;
    }
    brelse(bp);
    bfree(ip->dev, ip->addr[NDIRECT + 1]);
    ip->addr[NDIRECT + 1] = 0;
}
```

2 软链接

软链接的实现也非常的浅显，为了创建symbolic link file，我们只需要在sys_symlink中创建一个新的文件，并把目标路径，这一字符串，存到这个inode管理的第一个磁盘块即可。

```
//symlink in sysfile.c
ip = create(path, T_SYMLINK, 0, 0);
if(ip == 0){
    end_op();
    return -1;
}

// use the first data block to store target path.
if(writei(ip, 0, (uint64)target, 0, strlen(target)) < 0) {
    end_op();
    return -1;
}

iunlockput(ip);
```

不过为了让在使用open系统调用的时候忽略软链接文件本身的存在，那么我们应当专门为它创造一种文件类型。

```
#define O_NOFOLLOW 0x800
```

由于软链接文件可能同样指向另一个软链接文件，因此为了访问到它最终指向的实际文件，我们需要通过一个迭代的过程，直到遇到的新的inode不再是软链接文件类型才停止。为了防止出现死循环的情况，我们设定迭代的次数最多为10，代码如下所示：

```
struct inode*
follow_symlink(struct inode* ip) {
    uint inums[NSYMLINK];
    int i, j;
    char target[MAXPATH];
    for(i = 0; i < NSYMLINK; ++i) {
        inums[i] = ip->inum;
        // read the target path from symlink file
        if(readi(ip, 0, (uint64)target, 0, MAXPATH) <= 0) {
            iunlockput(ip);
            printf("open_symlink: open symlink failed\n");
            return 0;
        }
        iunlockput(ip);

        // get the inode of target path
        if((ip = namei(target)) == 0) {
            printf("open_symlink: path \"%s\" is not exist\n", target);
            return 0;
        }
        for(j = 0; j <= i; ++j) {
            if(ip->inum == inums[j]) {
                printf("open_symlink: links form a cycle\n");
                return 0;
            }
        }
        ilock(ip);
        if(ip->type != T_SYMLINK) {
            return ip;
        }
    }
    iunlockput(ip);
    printf("open_symlink: the depth of links reaches the limit\n");
    return 0;
}
```

3 参考材料

[1] Cox, R., Kaashoek, F., & Morris, R. (2022). *xv6: a simple, Unix-like teaching operating system*. Retrieved from <https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf>

[2] MIT. (2024). *Lab: file system*. Retrieved from <https://pdos.csail.mit.edu/6.1810/2024/labs/fs.html>