DESIGN

Yuqi Zhou, Lintao Lu

1. Summary

We use “open()” to create a normal file. If successful, “open()” will creates a file and return a file descriptor to us. If we take a look at the inode structure, we can find there is a variable called addrs. This vaiable contains pointers to disk blocks. If we want to creat a small file whose size is smallar than 64 bytes, instead of using addrs to store some pointers, we can just save the small file data here into inode.

```
12  // in-memory copy of an inode
13  struct inode {
14    uint dev;           // Device number
15    uint inum;          // Inode number
16    int ref;            // Reference count
17    struct sleeplock lock; // protects everything below here
18    int valid;          // inode has been read from disk?
19
20    short type;         // copy of disk inode
21    short major;
22    short minor;
23    short nlink;
24    uint size;
25    uint addrs[NDIRECT+1];
26    //uchar data[64];
27  };
28
```

2. Open

First, we must modify the open system call. As it shown below, I defined a new flag called “O_SMALL”, if we want to create a small file (smaller than 64 bytes), we just pass the flag to the “open()” system call, so it will know we need a small file this time, instead of using addrs to store pointers, “open()” will store the file data into inode. More details about “open()” are shown in another document.

```
301   if(omode & O_CREATE){
302     if(omode & O_SMALL){
303       ip = create(path, T_SMALL, 0, 0);
304     }
305     else{
306     ip = create(path, T_FILE, 0, 0);
307     }
```

3. Create

As we know, “open()” create a file and does some basic initialization. Actually inside “open()”, there is a method called “create()”, and it is the “create()” system call does the job to create a new file. So we just simple add a few lines to the “create()” system call, so it could handle the new “T_SMALL” situation.

```
252   if((ip = dirlookup(dp, name, &off)) != 0){
253     iunlockput(dp);
254     ilock(ip);
255     if(type == T_FILE && ip->type == T_FILE)
256       return ip;
257     if(type == T_SMALL && ip->type == T_SMALL)
258       return ip;
259     iunlockput(ip);
260     return 0;
261   }
```

4. Write

Because normal files are stored in disk’s blocks, we must modify “wite()” system call so it can write data to the inode. As we can see, “write()” system call only parses user’s inputs and throws exception if those inputs are illegal. So we don’t need to do any changes here. What’s interesting is “write()” calls “filewrite()” method, however, we also don’t need to change its code since “filewrite()” calls “writei()” method inside and it is the “writei()” function that really write data into inode. So we only need to change “writei()”, and make sure it can deal with “T_SMALL” situation. Notice, if user wants to write big file (bigger than 64 bytes), I just throws an exception here.

```
459   if(ip->type == T_DEV){
460     if(ip->major < 0 || ip->major >= NDEV || !devsw[ip->major].read)
461       return -1;
462     return devsw[ip->major].read(ip, dst, n);
463   }
464   if(ip->type == T_SMALL) {
465     //cprintf("Read from small file\n");
466     if (off > ip->size || off + n <off)
467       return -1;
468     if (off + n > ip->size)
469       n = ip->size - off;
470     //cprintf("off : %d\n", off);
471     for (int i = off; i<off + n; i++) {
472     memmove(dst + (i-off), ip->addrs + i%64, 1);
473     }
474     off = off + n;
475     //cprintf("off + n: %d\n", off);
476     return n;
477   }
```

5. Read

Traditional “read()” systme call can only read data form block. Just like “write()” system call, it also needs some modifications. “read()” system call is nearly the same as “write()”. It uses a method whose name is “readi()” to do the reading task. Just like write, I change codes in “readi()”.

```
459   if(ip->type == T_DEV){
460     if(ip->major < 0 || ip->major >= NDEV || !devsw[ip->major].read)
461       return -1;
462     return devsw[ip->major].read(ip, dst, n);
463   }
464   if(ip->type == T_SMALL) {
465      //cprintf("Read from small file\n");
466      if (off > ip->size || off + n <off)
467         return -1;
468      if (off + n > ip->size)
469         n = ip->size - off;
470      //cprintf("off : %d\n", off);
471      for (int i = off; i<off + n; i++) {
472      memmove(dst + (i-off), ip->addrs + i%64, 1);
473      }
474      off = off + n;
475      //cprintf("off + n: %d\n", off);
476      return n;
477   }
```

6. Other system call

Since “T_SMALL” is just a type of file, we don’t need to change other system calls suck like “unlink()”, “link()”. They work well.