Jeewon Ahn
Davvid Caballero
Friday, April 28
Programming Assignment Four

Documentation

Files Changed:

- Sysfile.c
- User.h
- Syscall.c
- Syscall.h
- Usys.S
- Makefile
- Ls.c
  - case T_SFILE:
  - case T_SDIR:
- Fcntl.h
  - #define O_SFILE 0x400 *1*
- Stat.h
  - #define T_SDIR 4 *1*
  - #define T_SFILE 5 *1*
- Fs.c

writei() function

```
memmove((char *)(ip->addrs) + off, src, n);
```

readi() function

```
memmove(dst, (char *)(ip->addrs) + off, n);
```

Files Added:

- Test1.c
  - Test creating small file and small directory
- Test2.c
  - Test creating a regular file in small directory
- Test3.c
  - Test creating a small file in regular directory
- Man mkSmallFilesdir
  - Man for mkSmallFilesdir and how to create small file.
- README.md

First I added new definitions for the small file type and small directory type(see *1*). Then I went through all of sysfile.c and fs.c that had T_DIR and T_FILE and duplicated it for T_SDIR and T_SFILE.

Second I made the mkSmallFilesdir() system call. It is an exact copy of mkdir() except for in create replace T_DIR with T_SDIR.

```
558  int
559  mkSmallFilesdir(void)
560  {
561    char *path;
562    struct inode *ip;
563
564    begin_op();
565    if(argstr(0, &path) < 0 || (ip = create(path, T_SDIR, 0, 0)) == 0){
566      end_op();
567      return -1;
568    }
569    iunlockput(ip);
570    end_op();
571    return 0;
572  }
```

In sysfile.c - sys_open()

```
int
sys_open(void)
{
  char *path;
  int fd, omode;
  struct file *f;
  struct inode *ip;

  if(argstr(0, &path) < 0 || argint(1, &omode) < 0)
    return -1;

  begin_op();


  if(omode & O_CREATE){
    if(omode & O_SFILE){
      char name[DIRSIZ];
      struct inode *dp = nameiparent(path, name);

      if(dp->type != T_SDIR){
        cprintf("NONONO");
        end_op();
        return -1;
      }
      ip = create(path, T_SFILE, 0, 0);
      if(ip == 0){
        end_op();
        return -1;
      }

    } else {
      char name[DIRSIZ];
      struct inode *dp = nameiparent(path, name);
      if(dp->type == T_SDIR){
        cprintf("nononon");
        end_op();
        return -1;
      }
      ip = create(path, T_FILE, 0, 0);
      if(ip == 0){
        end_op();
        return -1;
      }
    }
  } else {
    ip=namei(path);
    if(ip == 0){
      end_op();
      return -1;
    }
    ilock(ip);
    if(ip->type == T_DIR && omode != O_RDONLY){
      iunlockput(ip);
      end_op();
      return -1;
    }
    if(ip->type == T_SDIR && omode != O_RDONLY){
      iunlockput(ip);
      end_op();
      return -1;
    }
  }



  if((f = filealloc()) == 0 || (fd = fdalloc(f)) < 0){
    if(f)
      fileclose(f);
    iunlockput(ip);
    end_op();
    return -1;
  }
  iunlock(ip);
  end_op();

  f->type = FD_INODE;
  f->ip = ip;
  f->off = 0;
  f->readable = !(omode & O_WRONLY);
  f->writable = (omode & O_WRONLY) || (omode & O_RDWR);
  return fd;
}
```

sys_open() is changed to accept Small directories and small files. Also checking to make sure a small file doesn't get added to a normal directory and a regular file doesn't get added to a small directory.

Sysfile.c - create()

```
263  static struct inode*
264  create(char *path, short type, short major, short minor)
265  {
266    uint off;
267    struct inode *ip, *dp;
268    char name[DIRSIZ];
269
270    if((dp = nameiparent(path, name)) == 0)
271      return 0;
272    ilock(dp);
273    if((ip = dirlookup(dp, name, &off)) != 0){
274        iunlockput(dp);
275        ilock(ip);
276        if(type == T_FILE && ip->type == T_FILE){
277          return ip;
278        }
279        if(type == T_SDIR && ip->type == T_SDIR)
280          return ip;
281        if(type == T_SFILE && ip->type == T_SFILE)
282          return ip;
283          cprintf("here8\n");
284        iunlockput(ip);
285        return 0;
286      }
287
298    if(type == T_DIR){  // Create . and .. entries.
299      dp->nlink++;  // for ".."
300      iupdate(dp);
301      // No ip->nlink++ for ".": avoid cyclic ref count.
302      if(dirlink(ip, ".", ip->inum) < 0 || dirlink(ip, "..", dp->inum) < 0)
303        panic("create dotsz");
304    }
305
306    if(type == T_SDIR){  // Create . and .. entries.
307      cprintf("here2\n");
308      dp->nlink++;  // for ".."
309      iupdate(dp);
310      // No ip->nlink++ for ".": avoid cyclic ref count.
311      if(dirlink(ip, ".", ip->inum) < 0 || dirlink(ip, "..", dp->inum) < 0)
312        panic("create dots");
313    }
```

These are the main part of the lab. For open() read() and write() system call they all go to these functions. This is where we check if it is a small file so we can open or edit it into the inode.

Fs.c - readi()
memmove(dst, (char *)(ip->addrs) + off, n);

Fs.c - writei()
memmove((char *)(ip->addrs) + off, src, n);

Tests

We have 3 test files, test1.c, test2.c, and test3.c. Test1 tests open read and write for small directories and files. Test 2 tests that a normal file cannot be created in a small directory. Test3 tests and a small file cannot be created in a normal directory.

*Test 3 only works on a clean build after the other tests are run test3 mkdir fails due to not being able to create a normal directory in small directory.

```
$ test1
Make mkSmallFilesdir
        succeeded
Change to mkSmallFilesdir
        succeeded
Create small file
        succeeded
Writing to file
        succeeded
Closing file
        succeeded
Opening file
        succeeded
Reading file
        succeeded
Closing file
        succeeded
small file tests ok
$ test2
-----------------------
--Creating Normal File--
---in small directory---
-----------------------
cannot create/open this file in Small directory
error: create normal file failed!
```

```
$ test3
-------------------------
---Creating Small File---
---in normal directory---
-------------------------
cannot create/open this small file in this directory
error: create normal file failed!
```