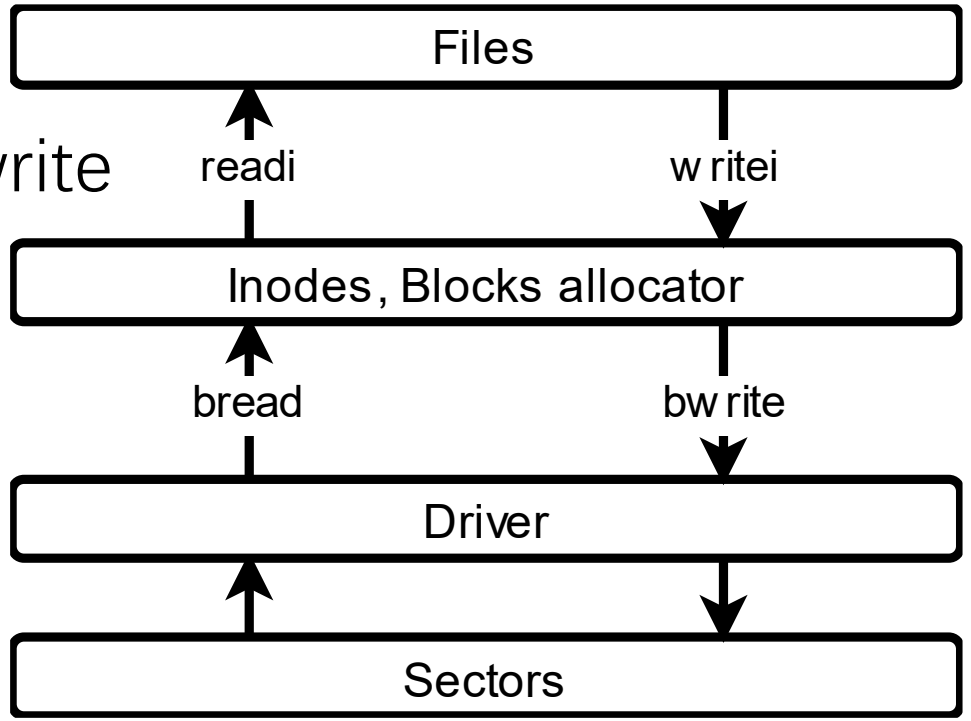
A simple file system

w1049

Overview

- Layer
- Sectors in disk
- Communicate with disk via bread/bwrite
- For step2: memcpy()
- For step3: send()/recv()
- readi/writei



Disk layout

- [superblock | inode blocks | free bit map | data blocks]
- Super block: 256 Bytes at most, #0, initialized when formatting
- magic: check if formatted

```
struct superblock {
    uint magic;
    uint size;
    uint nblocks;
    uint ninodes;
    uint inodestart;
    uint bmapstart;
} sb;
```

- Inode blocks: centralized storage
- Free bit map: cover all blocks
- Data blocks: indirect blocks, true data blocks

Inode

- Exactly divides 256 Bytes
- Type: File? Directory? Empty?
- Uid & Mode: Permissions
- Size & Blocks: Lazy deletion
- 10 direct + single / double indirect

- Directory = Special file
- “.” and “..” are added when created

```
struct dinode {
    ushort type : 2;
    ushort mode : 4;
    ushort uid : 10;
    ushort nlink;
    uint mtime;
    uint size;
    uint blocks;
    uint addrs[NDIRECT + 2];
};

struct dirent {  // 16 bytes
    uint inum;
    char name[MAXNAME];
};
```

Server & Client

- Modular
- Multi-client
- Permission check

```
struct clientitem {
    uint pwd;
    ushort uid;
};
struct clientitem *user;
```

- Multiprocess?
- I/O Multiplexing
- Event driven

```
while (1) {
        pool.ready_set = pool.read_set;
        pool.nready = select(pool.maxfd + 1, &pool.ready_set,
                             NULL, NULL, NULL);
        if (FD_ISSET(sockfd, &pool.ready_set)) {
            // handle new client
            int connfd = accept(sockfd, NULL, NULL);
            if (connfd < 0) err(1, ERROR "accept()");
            add_clients(connfd, &pool, client_init);
        }
        check_clients(&pool, serve);
    }
```

Command handler

- Easy to add new commands
- Just update the table
- Don’t need to modify the main loop

```
int NCMD = sizeof(cmd_table) / sizeof(cmd_table[0]);
for (int i = 0; i < NCMD; i++)
    if (strcmp(p, cmd_table[i].name) == 0) {
        ret = cmd_table[i].handler(p + strlen(p) + 1);
        break;
    }
```

```
static struct {
    const char *name;
    int (*handler)(char *);
} cmd_table[] = {
    {"I", cmd_i},
    {"R", cmd_r},
    {"W", cmd_w},
    {"E", cmd_e},
};
```

Makefile

- Generate dependency

```
SRC = $(wildcard *.c)
CC = gcc
CFLAGS += -Wall -Werror -fsanitize=address -g

all: fs disk client

fs: fs.o bio.o server.o client.o
    $(CC) $(CFLAGS) $^ -o $@

disk: disk.o server.o
    $(CC) $(CFLAGS) $^ -o $@

client: client.o clientmain.o
    $(CC) $(CFLAGS) $^ -o $@

include ${SRC:.c=.d}
%.o: %.c
    $(CC) $(CFLAGS) -c $< -o $@

%.d: %.c
    @set -e; rm -f $@; \
    $(CC) -M $(CFLAGS) $< > $@.$$$$; \
    sed 's,\($*\)\.o[ :]*,\1.o $@ : ,g' < $@.$$$$ >
$@; \
    rm -f $@.$$$$
```

Others

- Colorful shell

```
>>> print("File")
File
>>> print("\033[34m\33[1mDirectory\033[0m")
Directory
>>> print ("\33[1mTable Header\033[0m")
Table Header
```

- Log with macro

```
#define Log(format, ...)                                                \
    do {                                                                \
        fprintf(log_file, "[INFO] " format "\n", ##__VA_ARGS__); \
        fflush(log_file);                                               \
    } while (0)
```

Thank you!