CS111 Project 7: Unix V6 File System

Assignment Introduction

Isaac Scheinfeld

Unix V6 File System

Project Introduction

- History
  - Released 1975, first Unix to see widespread use
  - Small but powerful, with many modern features
  - Implemented in C, 16-bit machines
- Assignment
  - Used for many years in CS110
  - Provided with Unix V6 disk images, code to read sectors
  - Implement enough of the file system to read files

Unix V6 File System

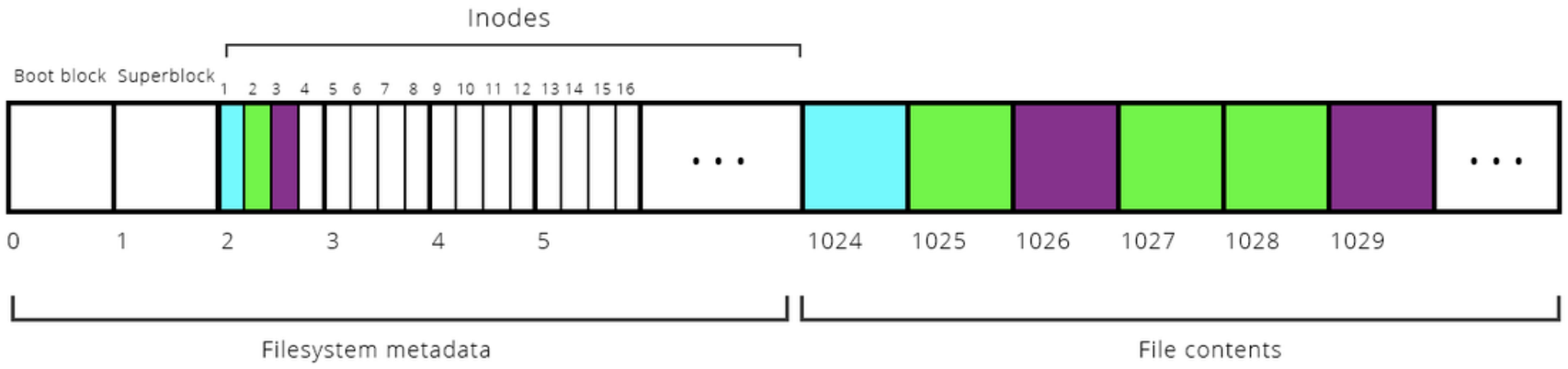
Code Structure

- Block layer (`diskimg.h/c`) - read and write sectors
- Inode layer (`inode.h/c`) - find inodes and block index lookup
- File layer (`file.h/c`) - read a block from a file
- Directory layer (`directory.h/c`) - directory lookup
- Pathname layer (`pathname.h/c`) - path lookup

Unix V6 File System

File System Structure - per disk

- Layout (`unixfilesystem.h`)
  - Block 0 - bootblock
  - Block 1 - superblock
  - Block 2 - inode list
  - Block 2 + `s_isize` - data
- Superblock (`filesys.h`)
  - First field `s_isize` is size in blocks of inode list



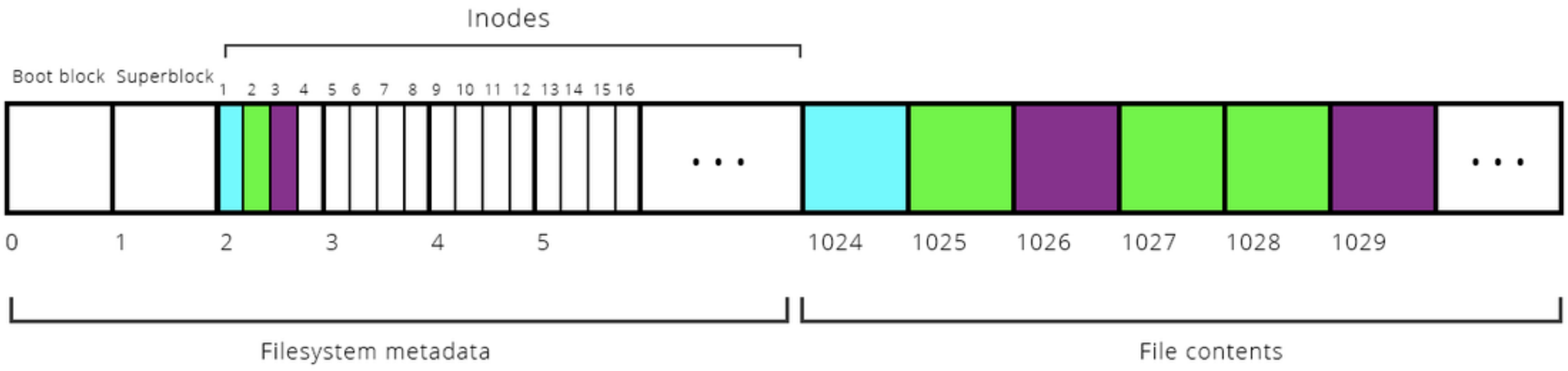
Unix V6 File System

File System Structure - per inode

- Inode struct (`ino.h`)
  - Size - 24 bits split across two fields `i_size0` and `i_size1`.
  - Blocks - inode contains array `uint16_t i_addr[8]`
    - The `i_mode` bit ILARGE determines index structure
    - False - each pointer is to a block
    - True - the first 7 pointers are indirect, the 8th is doubly-indirect
- Directory struct (`direntv6.h`)
  - Contains `uint16_t d_inumber` and `char d_name[14]`

Unix V6 File System

File System Structure



Inode 1 (stored in sector 2, offset 0):

Type: directory
Filesize: 32 bytes
Contents: 1024

Inode 2 (stored in sector 2, offset 32):

Type: regular file
Filesize: 1028 bytes
Contents: 1027, 1028, 1025

Contents of block 1024:

Bytes 0-15:	"a.mp3"	2
Bytes 16-31:	"b.txt"	3
	⋮	

Unix V6 File System

Code Structure

- Block layer (`diskimg.h/c`) - read and write sectors
- Inode layer (`inode.h/c`) - find inodes and block index lookup
- File layer (`file.h/c`) - read a block from a file
- Directory layer (`directory.h/c`) - directory lookup
- Pathname layer (`pathname.h/c`) - path lookup

Unix V6 File System

Code Structure

- Block layer (`diskimg.h/c`) - read and write sectors
  - Provided, reads 512-byte sectors, same as hardware interface
  - `int diskimg_readsector(int fd, int secNum, void *buf)`
- Inode layer (`inode.h/c`) - find inodes and block index lookup
- File layer (`file.h/c`) - read a block from a file
- Directory layer (`directory.h/c`) - directory lookup
- Pathname layer (`pathname.h/c`) - path lookup

Unix V6 File System

Code Structure

- Block layer (`diskimg.h/c`) - read and write sectors
  - `int diskimg_readsector(int fd, int secNum, void *buf)`
- Inode layer (`inode.h/c`) - find inodes and block index lookup
  - `int inode_iget(*fs, int iNum, struct inode *inp)`
  - `int inode_indexlookup(*fs, struct inode *inp, int blockNum)`
- File layer (`file.h/c`) - read a block from a file
- Directory layer (`directory.h/c`) - directory lookup
- Pathname layer (`pathname.h/c`) - path lookup

Unix V6 File System

Code Structure

- Block layer (`diskimg.h/c`) - read and write sectors
  - `int diskimg_readsector(int fd, int secNum, void *buf)`
- Inode layer (`inode.h/c`) - find inodes and block index lookup
  - `int inode_iget(*fs, int iNum, struct inode *inp)`
  - `int inode_indexlookup(*fs, struct inode *inp, int blockNum)`
- File layer (`file.h/c`) - read a block from a file
  - `int file_getblock(*fs, int iNum, int blockNum, void *buf)`
- Directory layer (`directory.h/c`) - directory lookup
- Pathname layer (`pathname.h/c`) - path lookup

Unix V6 File System

Code Structure

- Block layer (`diskimg.h/c`) - read and write sectors
- Inode layer (`inode.h/c`) - find inodes and block index lookup
- File layer (`file.h/c`) - read a block from a file
  - `int file_getblock(*fs, int iNum, int blockNum, void *buf)`
- Directory layer (`directory.h/c`) - directory lookup
  - `int directory_findname(*fs, const char *name, int dirNum, struct direntv6 *dirEnt)`
- Pathname layer (`pathname.h/c`) - path lookup

Unix V6 File System

Code Structure

- Block layer (`diskimg.h/c`) - read and write sectors
- Inode layer (`inode.h/c`) - find inodes and block index lookup
- File layer (`file.h/c`) - read a block from a file
  - `int file_getblock(*fs, int iNum, int blockNum, void *buf)`
- Directory layer (`directory.h/c`) - directory lookup
  - `int directory_findname(*fs, const char *name, int dirNum, struct direntv6 *dirEnt)`
- Pathname layer (`pathname.h/c`) - path lookup
  - `int pathname_lookup(*fs, const char *pathname)`

Unix V6 File System

Implementation Order

- Inode layer (`inode.h/c`) - find inodes and block index lookup
- File layer (`file.h/c`) - read a block from a file
  - **Inode tests should pass**
- Directory layer (`directory.h/c`) - directory lookup
- Pathname layer (`pathname.h/c`) - path lookup
  - **Pathname tests should pass**

Unix V6 File System

String Stuff - Equality Checking

- Filenames in directory entries are 14 characters “C strings”
  - Not null-terminated if it uses all 14 characters
- Two C string comparison functions
  - `int strcmp(const char *str1, const char *str2)`
  - `int strncmp(const char *str1, const char *str2, size_t num)`
- Return value
  - `< 0` : the first character that does not match has a lower value
  - `== 0` : the strings are equal
  - `> 0` : the first character that does not match has a higher value

Unix V6 File System

String Stuff - Tokenization

- Parsing pathnames requires tokenizing paths into names
  - Two common approaches, using `strsep` or `strtok`
- `char *strtok(char *str, const char *delim)`
  - `*str` : pointer to front of C string
  - `*delim` : C string of delimiters (i.e. “/“)
- `char *strsep(char **strp, const char *delim)`
  - `**stringp` : double pointer to front of C string
  - `*delim` : C string of delimiters

Unix V6 File System

String Stuff - Tokenization

- `char *strtok(char *str, const char *delim)`
  - `*str` : pointer to front of C string
  - `*delim` : C string of delimiters (i.e. “/“)

```
char str[] ="/a/path/of/chars";
char * token;
token = strtok (str, "/");
while (token != NULL) {
  printf ("%s\n", token);
  token = strtok (NULL, "/");
}
```