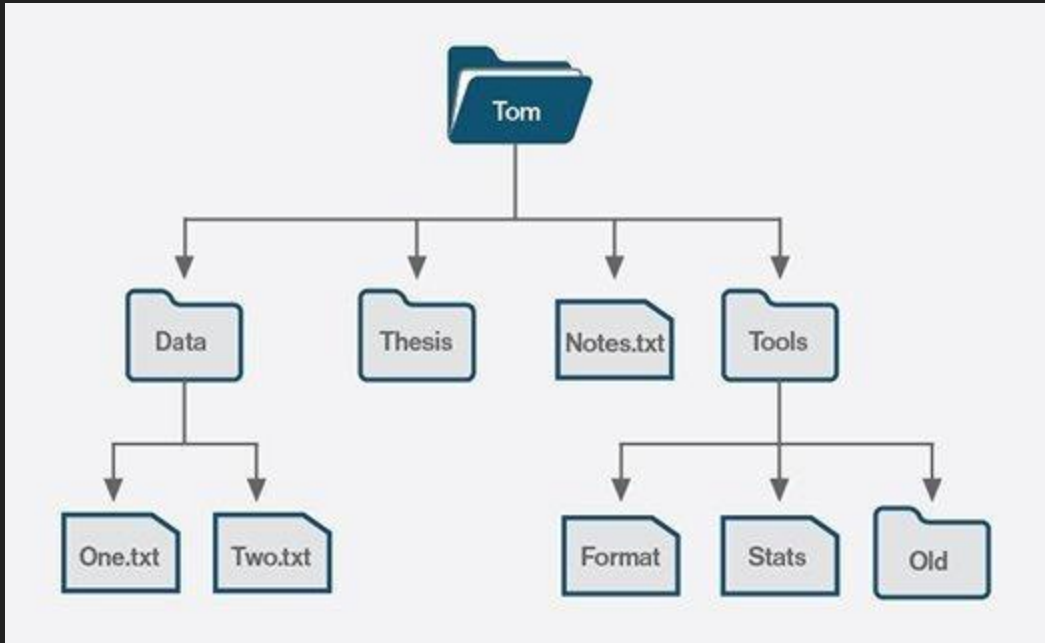
CSCI 452 Presentation

Raymond Healy, Koen Komeya, Ian Campbell

What we set out to accomplish

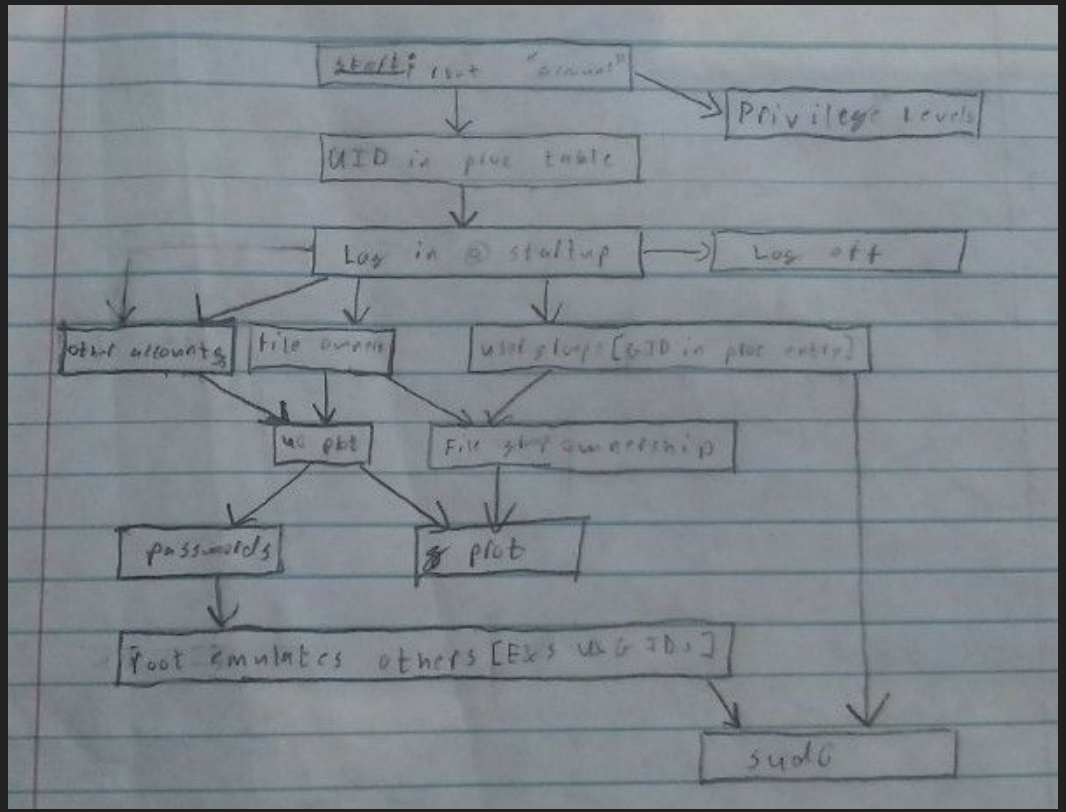
- Permission system - Raymond
- File System - Ian & Raymond
- Disk Driver - Koen



Permission System - Overview

- Generally designed to ape unix divided its permissions
  - Permissions divided into 3 rings, the owner, the owning group, and others
  - Permissions at each level are also divided
    - Read/Write (files, FS traversal)
    - Meta (change file ownership, location, and permissions)
    - Syscall execution (e.g. only root, sudoers, and owners kill a proc)
- When trying to create a plan I tried to think through logical extensions of existing elements.
  - E.g. “Protections based on UID” to “Changing UID should be restricted to root” to “the login shell must be spawned as root”

Permission System



Permission System - Early progress

```c
typedef struct pcb_s {
    // four-byte values

    // Start with these eight bytes, for easy access in assembly
    context_t *context;     // pointer to context save area on stack
    stack_t *stack;         // pointer to process stack

    inode_id_t wDir;        // ID of the working directory's inode

    int32_t exit_status;    // termination status, for parent's use
    event_t event;          // what this process is waiting for

    // two-byte values
    pid_t pid;              // unique PID for this process
    pid_t ppid;             // PID of the parent

    gid_t gid;              // group ID of this process
    uid_t uid;              // user ID of this process

    // one-byte values
    state_t state;          // current state (see common.h)
    prio_t priority;        // process priority (MLQ queue level)

    uint8_t quantum;        // quantum for this process
    uint8_t ticks;          // ticks remaining in current slice

    fd_t files[MAX_OPEN_FILES]; // File descriptors
} pcb_t;
```

```
(033) Is this land made for you and me?
(000)
(031) Nobody living can ever stop me,
(037) As I go walking that freedom highway;
(040) Nobody living can ever make me turn back
(035) This land was made for you and me.

$ test 1 1
M User 1.0 reports uid 100
M User 1.0 reports gid 0

$ list 0
        Baseline tests (bank 0):
                A, B, C: Print out name a finite nr of times w/ delays
                   D, E: Like A-C, but also checks write's return
                   F, G: Report, sleep (10 and 20 secs) and exit
                      H: Test orphan reparenting
                      I: Test killing children
                      J: Test overflow proc table
                   K, L: Iterate spawning user X and sleeping
                   M, N: Test various syscalls through children
                      P: Iterates reporting time and sleeping
                      Q: Tests a bad syscall
                      R: Echo SIO forever (NO EXIT!!!)
                      S: Sleep forever (NO EXIT!!!)
                   T, U: Spawn user W copies and then wait or kill them
                      V: Play with process priority

$ test 0 F
FFF
```

Permission System - Later progress

- Following this I largely shifted focus to FS until it was nearing completion
- At that point, I returned to my earlier work
  - Improved what was already there (group selection, logon with names and passwords)
  - Extended the reach of permissions forward (e.g. implementing file permissions)
- In testing these modifications (and with a fuller drive), I also ran into and helped to fix many of the bugs remaining in FS, as a bonus.

```
=======================================
Enter your username: Ray
Enter your password:

Test shell started
Try help for a list of commands

$ ls

    dwrwrwr    00  00    ./
    dwrwrwr    00  00    ../
    ---wrwr    00  00    .shadow
    -wrwrwr    00  00    testLongName
    --rwrwr    00  00    .groups

$ test 2 3
Test FS 3.0: Opened "/test.txt" as channel 2

Editing the new file by writing to it
Test FS 3.0: Closed "/test.txt"
(025) HELLO WORLD IN A NEW FILE

$ ls

    dwrwrwr    00  00    ./
    dwrwrwr    00  00    ../
    ---wrwr    00  00    .shadow
    -wrwrwr    00  00    testLongName
    --rwrwr    00  00    .groups
    -wrwrwr    64  00    test.txt

$
```

File System - Overview

- Very basic system to read and write to a device
- Reads and writes 512 byte blocks
- Currently functions on a RAMdisk and a disk driver emulated via QEMU.
- Reading, writing, opening, closing, creation, moving, and removal of files possible!
  - Done with Syscalls
  - Repurpose given read/write
- Each PCB contains array of File Descriptors
- Can do basic CAT and LS style operations.

```c
/*
 * This structure will contain a variety of data
 * relating to a file/directory in the FS.
 */
typedef struct inode_s {
    // Meta Data (16 bytes)
    inode_id_t id;
    uint32_t nBlocks;
    uint32_t nBytes;  // Either number of file bytes or number of subdirectories
    uint32_t nRefs;  // Number of inodes referencing this one

    // Permission information (8 bytes)
    uint32_t permissions: 24;
    uint32_t nodeType: 8;
    uid_t uid;
    gid_t gid;

    // Lock + Padding (4 bytes)
    uint8_t lock; // 1 byte
    uint8_t pad[3]; // 3 bytes

    // Indirect Pointers 4 bytes
    block_t extBlock; // Points to a block

    // Direct Pointers - each point to a block
    data_u direct_pointers[NUM_DIRECT_POINTERS]; // 32 + 14 * 16 = 256 bytes per inode
} inode_t;
```

```c
#define NUM_DIRECT_POINTERS 14

typedef uint32_t block_t;

typedef struct {
    uint32_t devID : 8;
    uint32_t idx : 24;
} inode_id_t;

typedef struct {
        char name[12];
        inode_id_t inode;
} dirEnt_t;

// Each is 16 bytes
typedef union {
    block_t blocks[4];
    dirEnt_t dir;
} data_u;
```

File System - Philosophies

- Keep it simple
  - Designed inode early
  - Identified block sizes and inode sizes - 2 inodes per block
  - Don’t get fancy - currently no extension blocks - can only have 14 direct blocks
- Test early & plug and play
  - RAMdisk
    - Allows us to test everything other than reboot persistence
      - Blessing in disguise
  - Make a device interface standard for all disk drivers
    - Register
    - Fill a structure with two function pointers (read & write)

File System - Issues we had

- Unintended recursion - stack smashing on invaild read
- Write is far more complicated than read
- Allocating and freeing blocks has many points of failure
- To get a basic FS working requires a lot of helper functions to even be possible
  - Alloc/Free Blocks, Alloc/Free inodes, getSubDir, rmDirEnt, addDirEnt, etc...

File Systems - If we had more time

- Extension Blocks
- FS Caching scheme

Disk Driver - The Final Product

- PATA Driver
- Based on PATA revision 6,0
- Compatible with Multiple Disks
- Works on 512 byte sectors
- In theory, supports both hard disk drives and solid state drives,

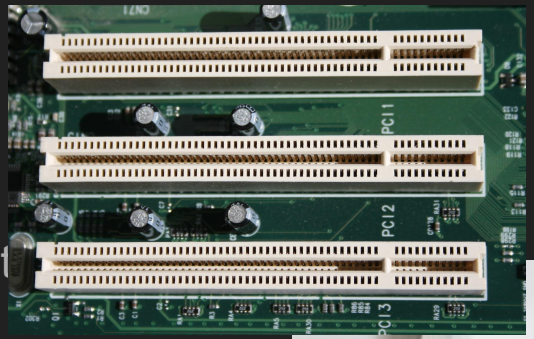
```
for (int ch = 0; ch < 2; ch++) {
    for (int dr = 0; dr < 2; dr++) {
        // Select drive
        _ide_write(ch, ATA_REG_HDDEVSEL, 0xA0 | (dr << 4));
        _sleep1ms();

        // Identify command
        _ide_write(ch, ATA_REG_COMMAND, ATA_CMD_IDENTIFY);
        _sleep1ms();

        // Ensure this is ATA
        if (_ide_read(ch, ATA_REG_STATUS) == 0)
            continue;
        int fail = 0;
        unsigned char status;
        do {
            status = _ide_read(ch, ATA_REG_STATUS);
            if ((status & ATA_SR_ERR)) {fail = 1; break;}
        } while ((status & ATA_SR_BSY) || !(status & ATA_SR_DRQ));
        if (fail)
```

Disk Driver - Overview

- First, talk to the PCI (Peripheral Component Interconnect) bus to get the PATA devices.
- PATA stands for Parallel AT Attachment, where AT referred to an IBM PC model, short for “Advanced Technology”
- Based on Western Digital’s IDE, which stands for Integrated Drive Electronics. Because of their development alongside each other, they are used interchangeably to refer to the same thing.
- PATA is basically one of the protocols for communicating with the device that interfaces with the actual hard disk.



Disk Driver - Issues I faced

- Was hoping to implement AHCI but realized I probably wouldn’t be able to finish it, so I switched course to PATA.
- Didn’t work while testing due to misconfigured QEmu. Solution - include the hardware components in the command line, but only what you need.
- Mysterious interrupts firing on the Hard Disk ISR even when interrupts were disabled.

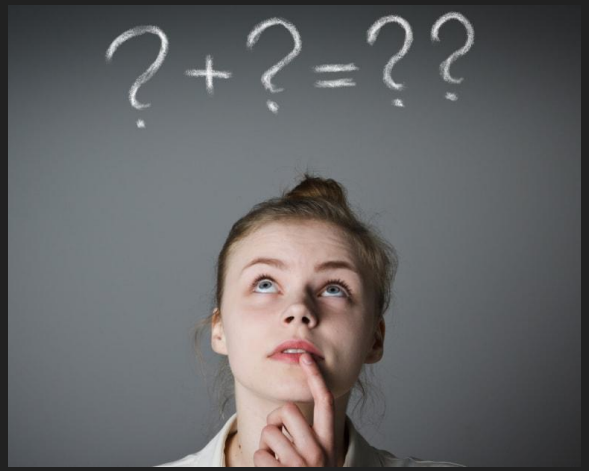


Image Sources

https://i.ebayimg.com/images/g/kSUAAOSwz71ZSomJ/s-l300.jpg
https://target.scene7.com/is/image/Target/GUEST_d9d776ab-1160-469d-918e-2fa3666c4d9a?wid=488&hei=488&fmt=pjpeg
https://cdn.ttgtmedia.com/rms/onlineImages/TT_tree_mobile.jpg
https://images2.minutemediacdn.com/image/upload/c_crop,h_1126,w_2000,x_0,y_122/f_auto,q_auto,w_1100/v1554930340/shape/mentalfloss/536399-istock-483928994.jpg
https://upload.wikimedia.org/wikipedia/commons/6/67/PCI_Slots_Digon3.JPG
https://www.itprotoday.com/sites/itprotoday.com/files/styles/article_featured_retina/public/Hard%20disk%20drive.png?itok=ggp_kRdk