持久化：数据的故事

蒋炎岩

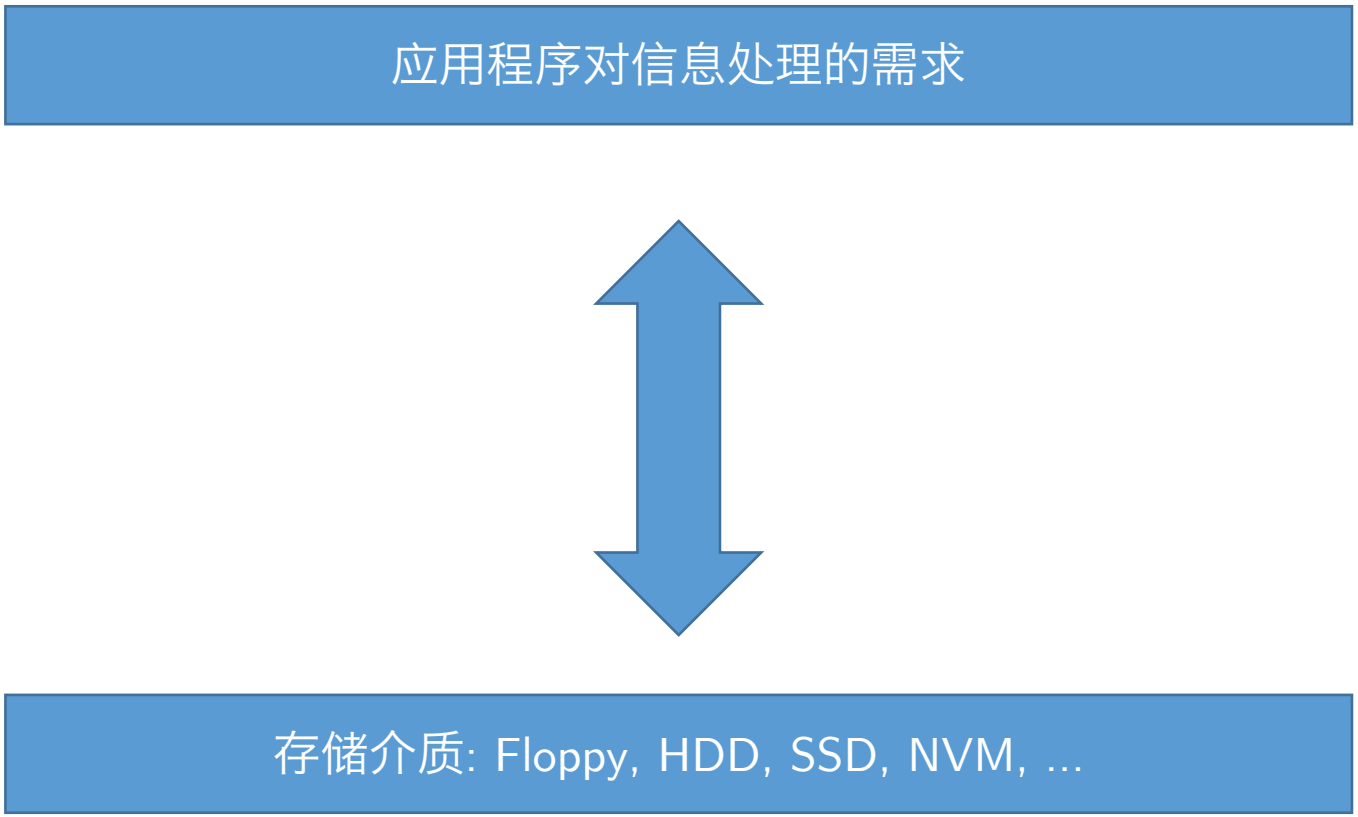
南京大学 | 计算机软件研究所 | 系统与软件分析研究组

数据的故事

- 计算机为了完成人类世界里的计算任务，必须处理数据
  - 从介质中读取、保存到介质
- 中学生的学科奥林匹克竞赛(IMO, IPHO, ICHO, ...)中
  - IOI (International Olympiad in Informatics, since 1989)的名字是非常有趣的，“信息学”
- 问题：如何持久可靠地保存数据?
  - 操作系统中的“持久化”(persistency)问题

问题的提出

- 怎样“bridge the gap”？



存储器：历史

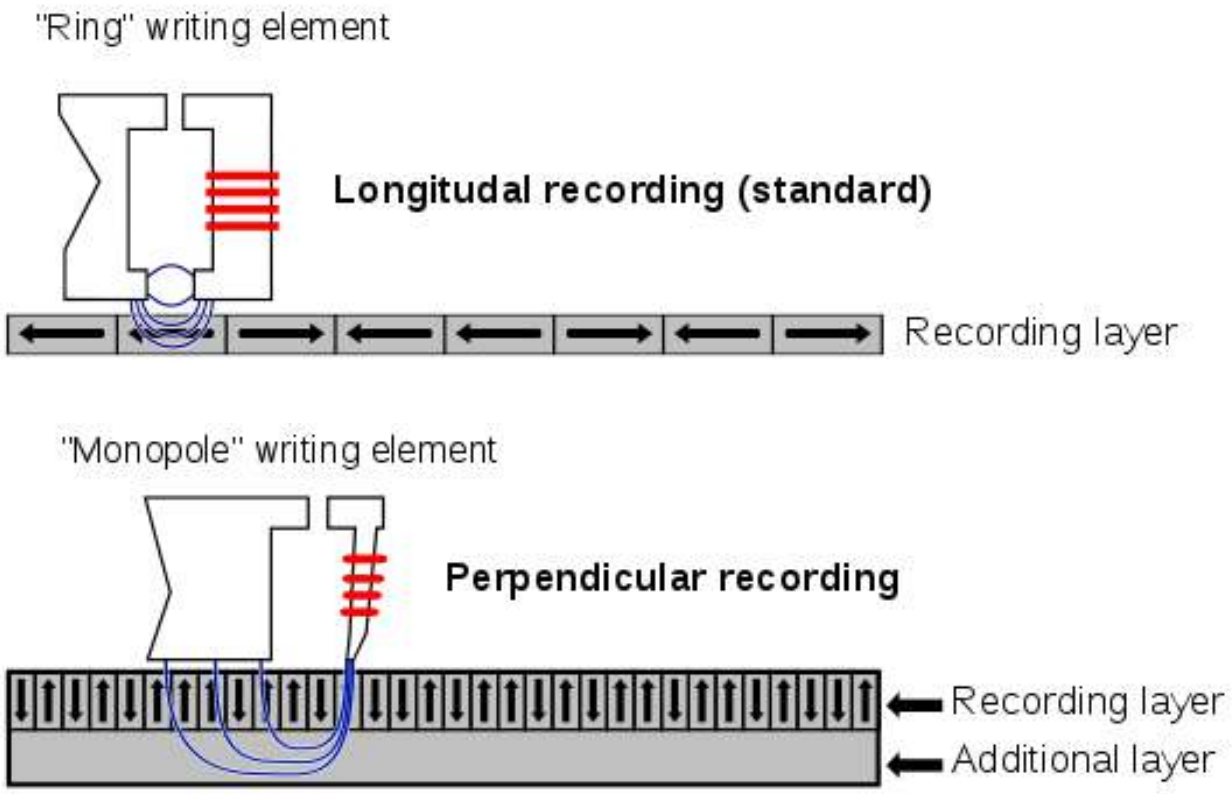
存储器是一种I/O设备

- 逻辑上说，是一个数组
  - 通过DMA方式传输数据
  - 传输完毕后以中断形式通知处理器

- 支持连续数据的高速读写
  - 随机读写性能会略慢一些

基于磁介质的存储设备

- 可磁化物质的磁场方向可以用来记录1bit信息



存储设备：磁带(1950s-)

- IBM726磁带机
  - 2.3MB, 450kg
- 磁带只支持顺序读写
  - 结构简单、容易制造

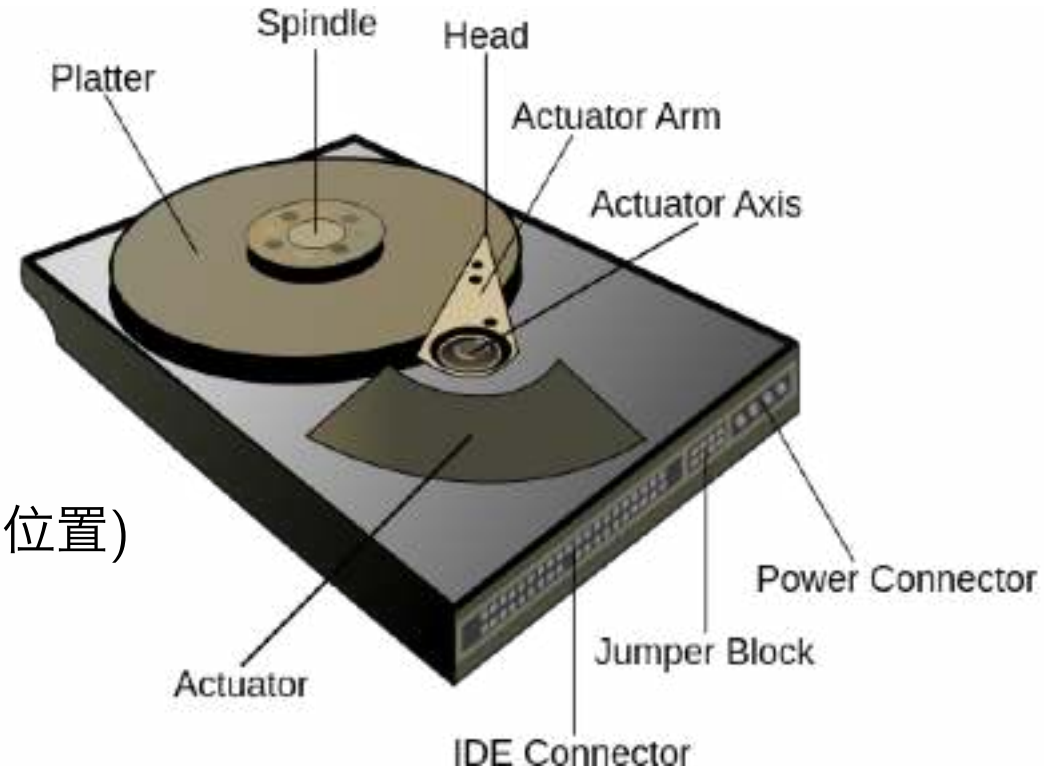
存储设备：软(磁)盘(1970s-1990s)

- 5.25英寸(软)软盘 (1976)
- 3.5英寸(硬)软盘 (1978, SONY)
  - 80磁道 * 18扇区 * 2面 * 512B

存储设备：硬(磁)盘(1960s-)

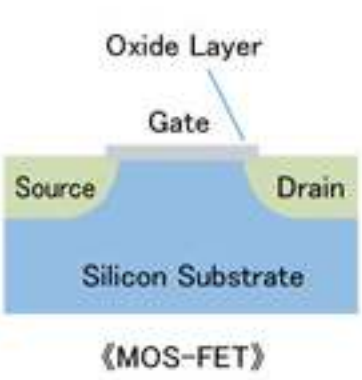
- 高速旋转的盘片
- 磁头扫过实现读/写

- 以扇区为单位访问
  - 转速越快，速度越快
  - 但延迟不固定(取决于磁头位置)

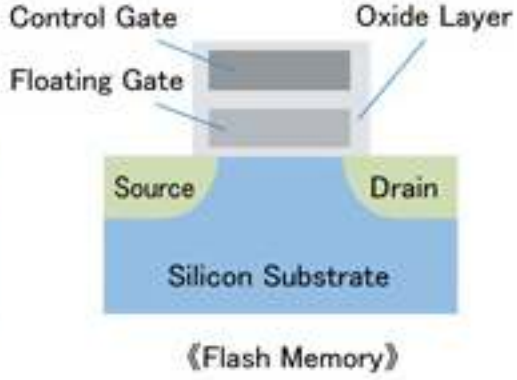


基于电介质的存储设备

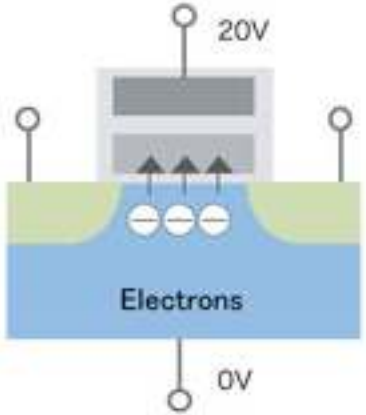
- NAND Flash
  - 对Floating Gate充/放电，实现1bit信息存储
- 3D XPoint
  - 改变材料的电阻实现1bit信息存储(黑科技?)



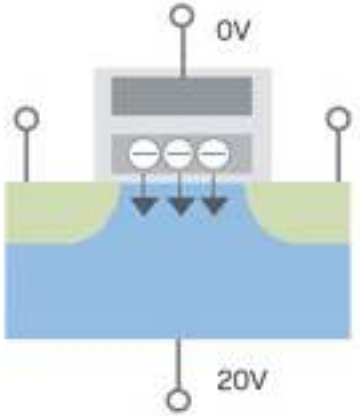
《MOS-FET》



《Flash Memory》



《Write operation》
A voltage applied to the control gate causes a tunnel current to flow through the oxide layer, thereby injecting electrons into the floating gate.



《Erase operation》
A voltage applied to the silicon substrate releases the electrons accumulated at the floating gate.

存储设备：Sold-State Drive (1990s-)

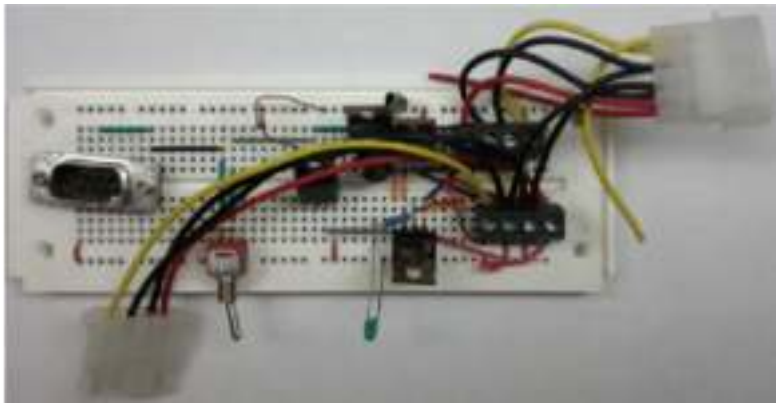
- 实现速度的飞跃(GB/s)
  - 更快的速度
  - 更低的延迟
  - 更高的可靠性

- 有了一个新名字：非易失性存储(non-volatile memory)
  - 我们已经习惯了“内存断电就消失”，但未来会是怎样?
  - NVM会怎样改变现在的memory hierarchy?

K. Bailey, L. Ceze, S. D. Gribble, and H. M. Levy. Operating System Implications of Fast, Cheap, Non-Volatile Memory. In *Proc. of HotOS*, 2011.

号外：新事物总有缺陷

- 13/15 SSDs都有缺陷
  - bit corruption
  - un-serializable writes
  - corruption
- 但比HDD好(dead device)



Device ID	Vendor -Model	Price ($/GB)	Type	Year	P?
SSD#1	A-1	0.88	MLC	'11	N
SSD#2	B	1.56	MLC	'10	N
SSD#3	C-1	0.63	MLC	'11	N
SSD#4	D-1	1.56	MLC	'11	-
SSD#5	E-1	6.50	SLC	'11	N
SSD#6	A-2	1.17	MLC	'12	Y
SSD#7	E-2	1.12	MLC	'12	Y
SSD#8	A-3	1.33	MLC	'11	N
SSD#9	A-3	1.33	MLC	'11	N
SSD#10	A-2	1.17	MLC	'12	Y
SSD#11	C-2	1.25	MLC	'11	N
SSD#12	C-2	1.25	MLC	'11	N
SSD#13	D-1	1.56	MLC	'11	-
SSD#14	E-1	6.50	SLC	'11	N
SSD#15	E-3	0.75	MLC	'09	Y

M. Zheng, J Tucek, F. Qin, and M. Lillibridge. Understanding the Robustness of SSDs under Power Fault. In *Proc. of FAST*, 2013.

文件系统

文件和目录：管理数据的自然方式

- 文件：一个有名字、可以读/写的数据对象
- 目录：文件的集合
- 文件和目录
  - 体现了局部性：相关的数据(文件)将位于相近的位置(目录)
  - 不关心数据本身：应用程序负责解读

文件系统即数据结构

- 一个文件系统是一个key-value mapping
  - 把文件路径映射到一个文件(可读、可写的字节序列)

- Keys形成一个树状的结构
  - 能获得目录下的子目录和文件
  - 能获得目录的父目录(.和..)

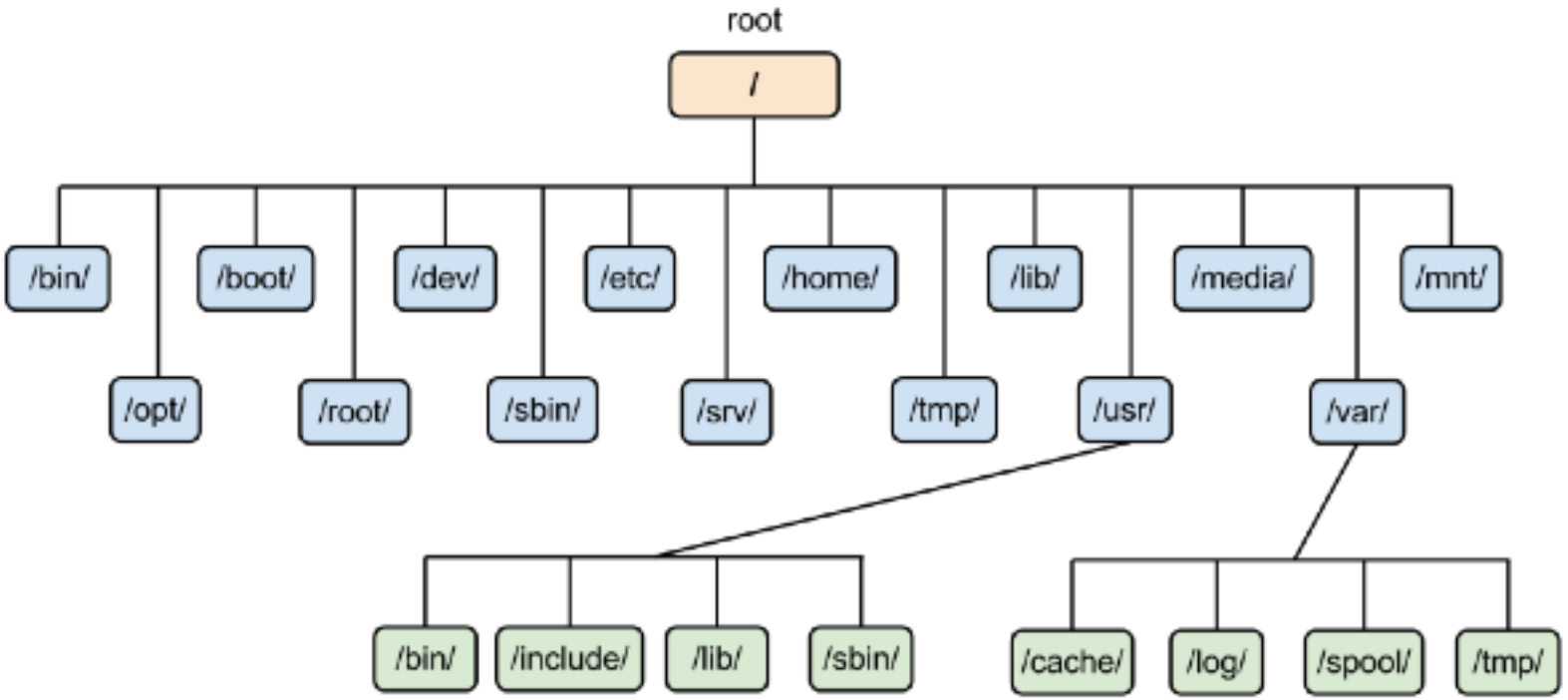
- 根据“当前工作目录(CWD, current working directory)”可以找到系统中的各种文件
  - ./a.out; /bin/bash; ...
  - 问题：chdir改变进程 or 线程的工作目录？为什么？

号外：如果线程要维护自己的工作目录?

- 如果你man 2 mkdir，会看到有mkdirat系统调用
  - 一系列目录操作的都有xxxxat的版本
  - 为需要的线程维护文件描述符(当前目录)

```
int openat(int dirfd, const char *pathname, int flags, mode_t mode);
int mkdirat(int dirfd, const char *pathname, mode_t mode);
int linkat(int olddirfd, const char *oldpath, int newdirfd,
           const char *newpath, int flags);
int unlinkat(int dirfd, const char *pathname, int flags);
```

Linux文件系统结构



多个文件系统实现并存

- 计算机中可以有多个不同的文件系统
  - 系统里可以有多个存储器(CD-ROM, HDD1, HDD2, SSD, ...)
  - everything is a file (procfs, devfs, sysfs, ...)

- 每个文件系统都是一个key-value mapping
  - 文件系统有自己的“根”(root) - /
  - 例如devfs: /input, /stdin, /random, /tty1, ...
  - 移动硬盘: /学习资料/letters-numbers.avi

- 文件系统可以被“挂载”(mount)到一个目录
  - /dev就是个普通的目录；把devfs的/拼接到/dev就好了

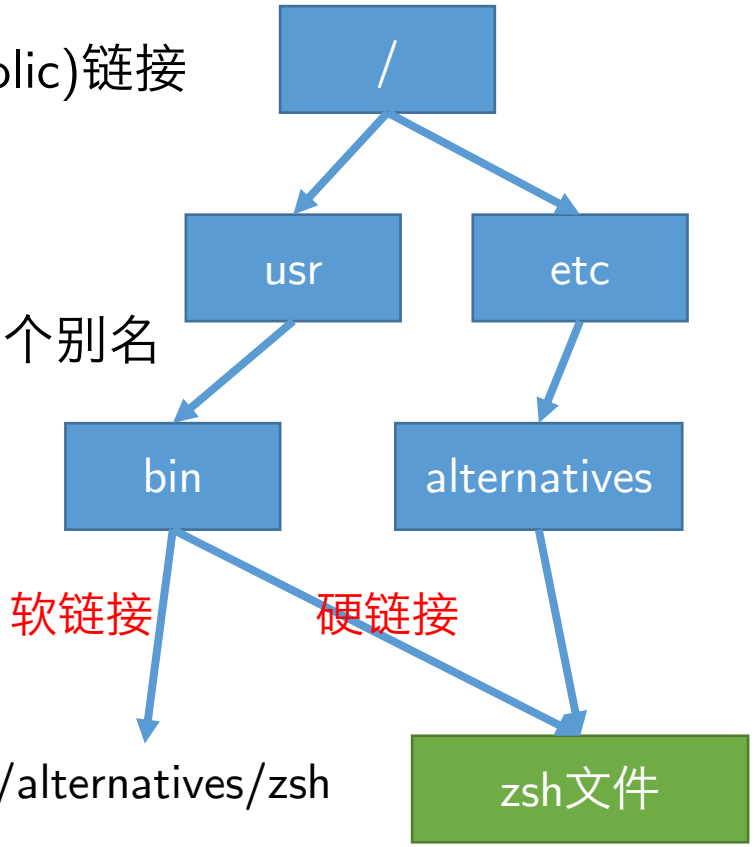
The Magical Mount

- Everything is a file, everything is in a file system
  - /dev, /proc, /mnt/cdrom0, ...
  - mount的有趣之处：mount挂载一个文件(everything is a file!)
    - 一个设备(优盘、磁盘.....)
    - 一个.iso镜像文件(以前在Windows上必须借助虚拟光驱软件)
    - 一个远程目录(网络目录、虚拟机共享目录、.....)

- 实现起来非常容易
  - 只需要修改路径解析部分的代码即可

The Magical Key-Value Mapping

- 是否允许多个key映射到同一个value?
  - 当然可以了：硬(hard)链接/软(symbolic)链接
- 硬链接
  - 在同一个文件系统中同一个文件的两个别名
  - 必须是同一个文件系统中的文件
- 软链接
  - 只是一个路径
  - 可以跨文件系统、可以是目录



链接的用处

- 类似于“快捷方式”

- 在需要别名的时候减少数据的副本
  - /bin/zsh → /etc/alternatives/zsh (symlink)
  - /usr/lib/libau.so → libau.so.2 (symlink)
  - /usr/bin/gcc → gcc-5 (symlink)
  - Online Judge: testcase1.in → prob.in (注意权限)

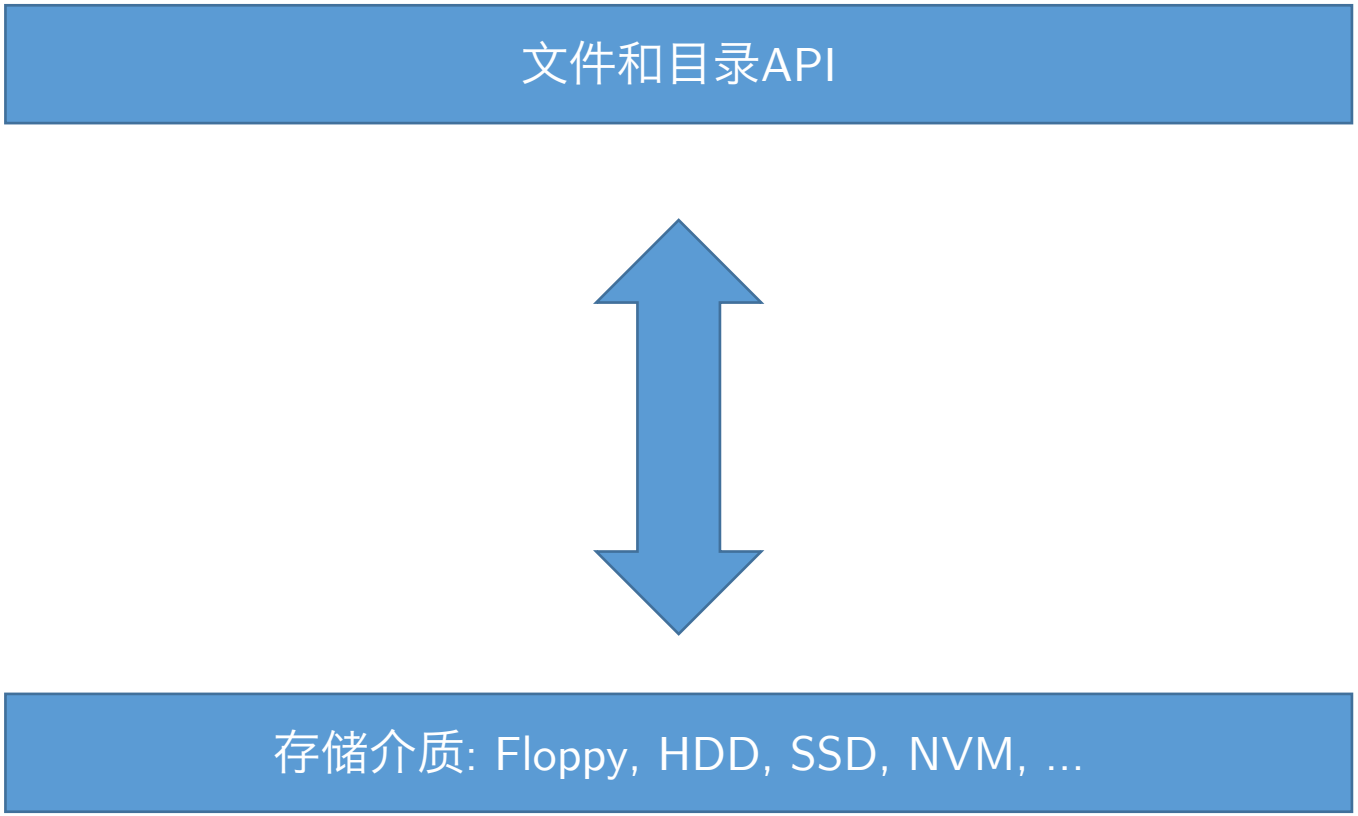
链接带来的趣事

- UNIX只允许对文件做硬链接
  - 无法区分两个链接的文件(这个世界很安全)
- 但符号链接不受此限制
  - `mkdir /tmp/a && ln -s /tmp/a/a /tmp/a`创造了一个环
  - `rm -rf ..`到底应该删除谁?
- 随着OS的功能不断增加，它的语义也越来越复杂 ☺

文件系统：实现

Bridging the Gap

- 文件系统：在存储介质上实现文件和目录API

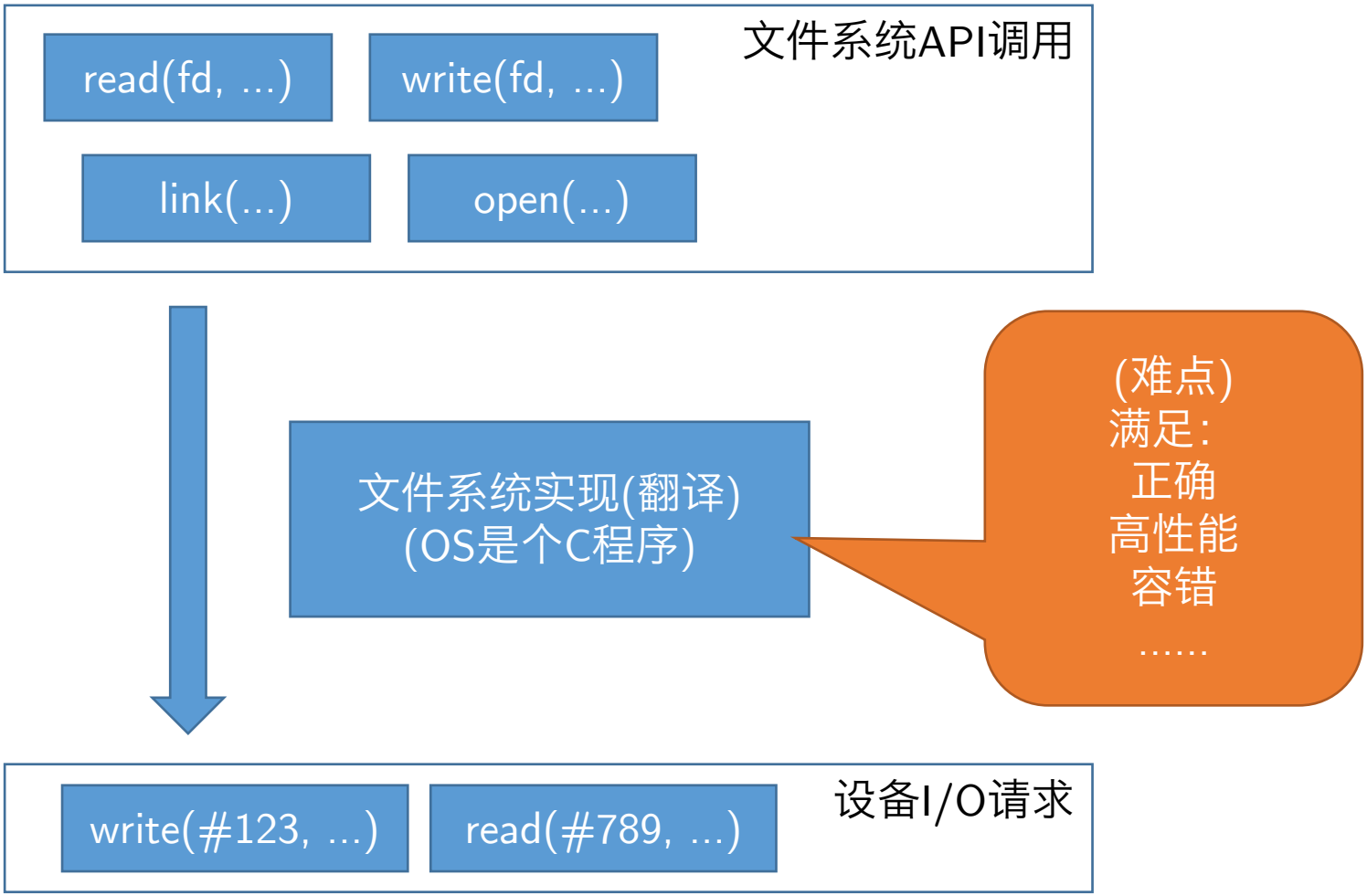


文件系统：实现

- procfs: 维护自己的目录结构(/cpuinfo, /[pid]/, ...)和文件内容
  - 不支持link, unlink, ... (你可以试着删除，sudo也删不掉)
  - 在Lab2里你们也会实现自己的procfs

- 存储器上的文件系统
  - 所有支持的操作都需要实现
  - 把文件系统操作(read, unlink, ...)翻译成设备的read/write

文件系统实现的本质


文件系统API调用
read(fd, ...)
write(fd, ...)
link(...)
open(...)
文件系统实现(翻译)
(OS是个C程序)
(难点)
满足:
正确
高性能
容错
.....
设备I/O请求
write(#123, ...)
read(#789, ...)

Key-Value Mapping：实现

- C++中map, unordered_map分别是用什么实现的？它们用来实现磁盘上的文件系统怎么样？
  - 需要支持的操作
    - 目录操作chdir, readdir, mkdir, link, unlink
    - 文件操作open, close, read, write, lseek
  - 底层存储器
    - 内存(ramdisk)、HDD、SSD、……
  - 应用场景
    - TB级的存储，无法全部保存在内存中

- 每个目录是一个map; 每个文件也是一个map