

# How do users interact with disks Storage?

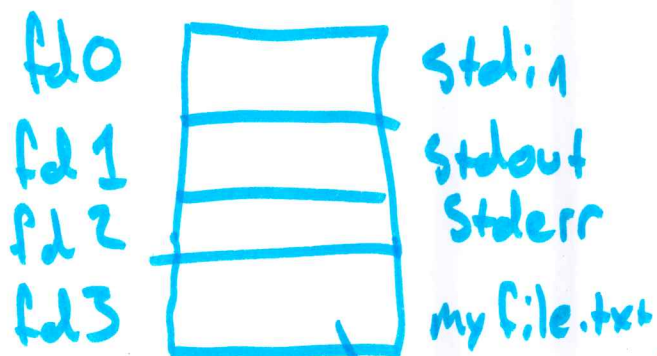
## reading/writing Files

```
int fd = open("myfile.txt", O_RDONLY);
```

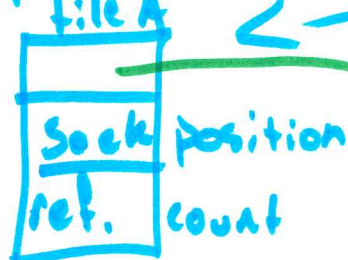
↓  
file descriptor → call to the operating system

→ index into a table in the OS

file descriptor table

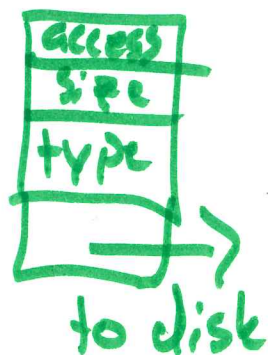


Open file table



Shared by all processes

V-node table



per-process basis

Process → ~~an~~ ~~applicati~~

the running code + other things  
that go with it

↓  
an OS

Construct

points to where  
file info is

File descriptor table → per process

Open file table → 1 in OS → dynamic  
info. per open  
file  
holds all open files

V-node table → holds all info about the  
file, including physical location

↑  
file system specific

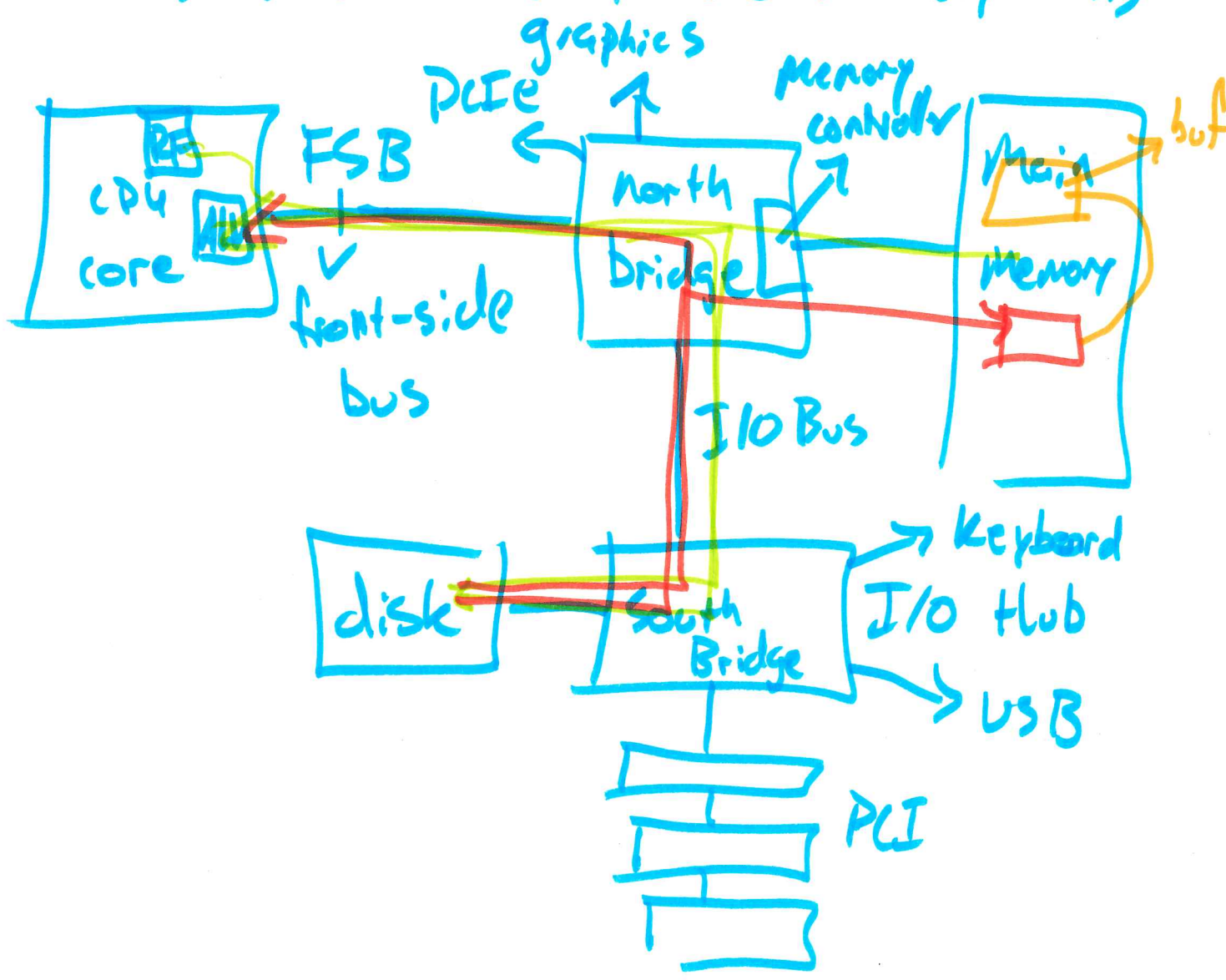
↳ static information  
for file

---

read(fd, buf, 100);

What actually happens?

1) read asks the OS to do it (syscall)



2) Request a sector from the disk

3) disk writes data to memory (DMA)

4) copy data into buf  
OS

Direct memory access



How to send ~~message~~ message to disk?

★ - Pick an address for I/O

↳ Memory-mapped I/O MMIO

~~PIO~~

- PIO - programmed I/O

inb } read/write  
outb }

```
while (STATUS == BUSY) {
```

```
    //spin
```

```
}
```

```
    read the data
```

→ spin loop

```
    movl ( ), %i  
    cmpl  
    jne
```

Called Polling the device

↳ may need a timeout

↳ inefficient

→ CPU can't do anything else

→ Waste lots of energy

Solution?

Interrupt the CPU

Interrupts asynchronously

change control flow

movl (%esp), %eax  
addl %ecx, %eax  
mull %ecx  
→ movl %eax, 8(%esp)

int. handler:

|||||  
|||||  
|||||  
|||||

int ret

Interrupt

Interrupts only happen @ inst. boundaries  
- precise interrupts