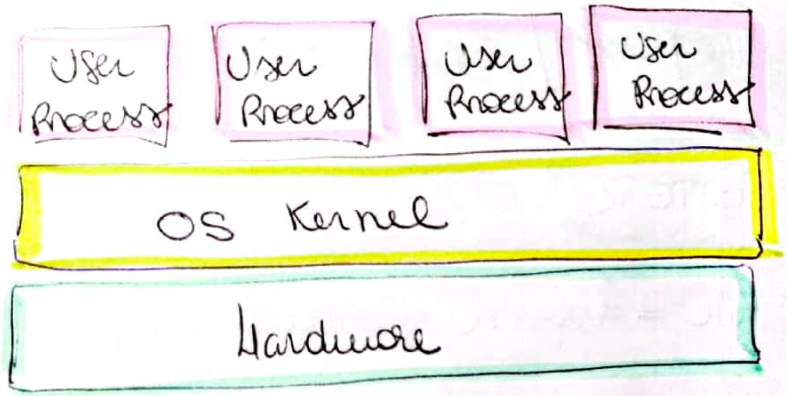
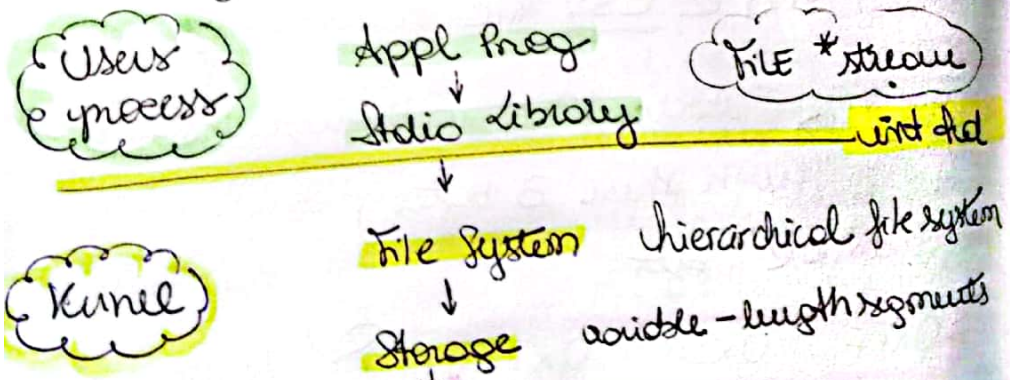
Syscalls de um buffer?

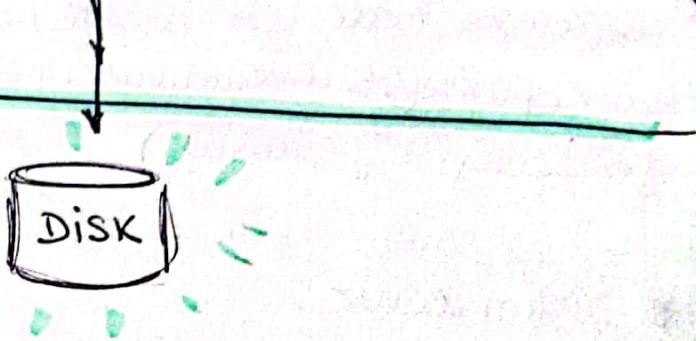
Typical ~~Operating~~ Operating System

- Protection & Security → make the machine safe.



→ Layers of Abstraction





System calls

- Kernel provided system services: "protected" procedure call.

fopen, fclose, printf, fgetc, getchar...

Appl Prog
↓
Stdio Library
↓
File System

User / Kernel

open, close, read, write, seek *

Obs: Unix has ~150 system calls!

→ Mechanism ←

- Processor modes:

- User mode: can execute normal instructions and access only user memory;

- Supervisor mode: can execute normal instructions, privileged instructions and access all of memory (e.g., devices)

System calls

- User cannot execute privileged instructions;
- Users must ask OS to execute these - system calls;
- System calls are OFTEN implemented using TRAPS (int);
- OS gains control through trap, switches to supervisor mode, performs service, switches back to user mode, and gives control back to user (iret).

System-call interface = ADTs

ADT
operations

FILE INPUT/OUTPUT
open, close, read, write, dup

PROCESS CONTROL
fork, exit, wait, kill, exec, ...

INTERPROCESS COMMUNICATION
pipe, socket, ...

open system call ①

INVOKE — open — possibly to create a file or device

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open(const char *pathname, int flags,
         mode_t mode);
```

mode is the permissions to use if file must be created

flags examples:
O_RDONLY
O_WRITE
O_CREATE

DESCRIPTION

The open() system call is used to convert a pathname into a file descriptor (a small, non-negative integer for use in subsequent I/O as with read, write, etc). When the call is successful, the file descriptor returned will be...
input/output

② close system call

NAME

close a file descriptor

SYNOPSIS

int close (int fd);

DESCRIPTION

closes a fd, so that it no longer refers to any file and may be reused. Any locks held on the file it was associated with, and owned by the process, are removed (regardless of the file descriptor that was used to obtain the lock)...

③ Read system call

- read from a file descriptor
- int read (int fd, void *buf, int count);

doesn't read lines, it just read bytes!

- read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.

If count is zero, read() returns zero and has no other results. If count is greater than SSIZE_MAX, the result is unspecified.

RETURN VALUE

On success, the number of bytes read is returned (zero indicates END of file), and the position is advanced by this number.

It's not an error if this number is smaller than the number of bytes requested ... on error, −1 is returned, and errno is set appropriately.

④ Write system call

saída de dados de 1 programa

- Write to a file descriptor
- Grava dados de 1 buffer declarado pelo usuário em 1 det. dispositivo (ex: arquivo)

```
int write(int fd, void *buf, int
          count);
```

writes up to count bytes to the file referenced by the fd from the buffer starting at buf.

RETURN VALUE

On success, the number of bytes written is returned (0 indicates NOTHING was written). It is not an error if this number is smaller than the number of bytes requested... On error, -1 is returned, and errno is set appropriately

Buffered Input / Output

(cont)

```
int getchar(void) {
    static char buf[1024];
    static char *p;
    static int n = 0;

    if (n--) return *p++;

    n = read(0, buf, sizeof(buf));
    if (n <= 0) return EOF;
    p = buf;
    return getchar();
}
```

Quant. exata de bytes em 1 Kilobyte. Este criando um buffer de 16KB por exemplo, ie 16 x 1024 !!!!

end of file? ✓

Conditions in a computer operating system where no more data can be read from a data source, the data source is usually called a file or stream.

fopen

FILE * fopen (char *name, char *rw){

- Use malloc to create a struct _iobuf;
- Determine appropriate "flags" from "rw" parameter;
- call open to get the file descriptor;
- Fill in the _iobuf appropriately

}

Stdio library

- fopen, fclose
- feof, ferror, fileno, fstat
- fflush
- fgetc, fgets, fread
- fputc, fputs, fwrite
- printf, fprintf
- scanf, fscanf
- fseek