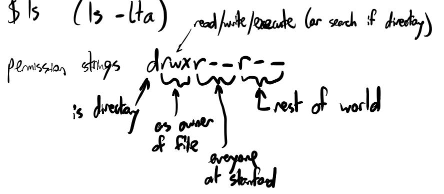
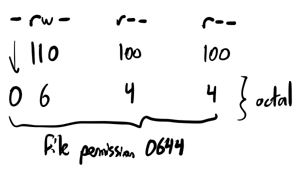
how OS allows us to programmatically interact w/ file system

$ ls   (ls -lta)

permission strings





$ cp (also a C-program shipped on Linux by default)

$ tee - takes stdin and writes to as many args (as files) that you want (also stdout)

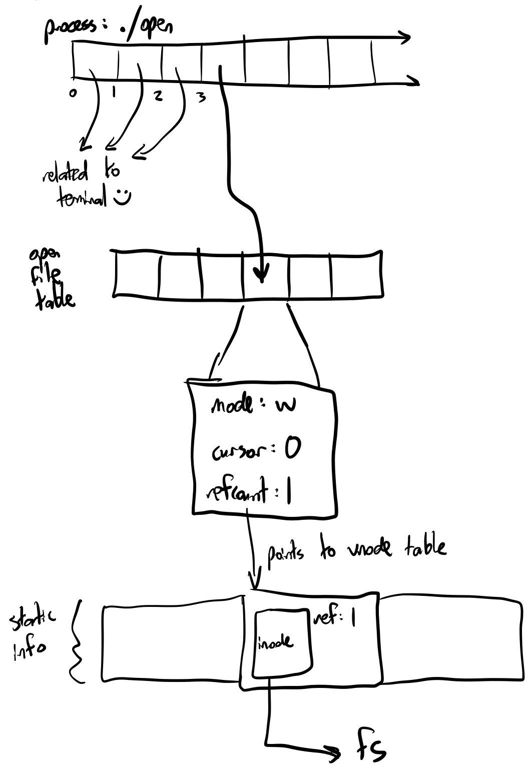
```
int open(const char *name, int flags);
int open(const char *name, int flags, int mode);
```
} system calls

```
int close(int fd);
```

open.c

```
int main(int argc, char *argv[]) {
    int fd = open("empty.txt",
                  O_CREAT | O_EXCL | O_WRONLY,
                  0644); ← octal #
    if (fd == -1) return 1;
    close(fd);
    return 0;
}
```

per-process basis: array called file descriptor table



more system calls:

```
ssize_t read(int fd, char buf[], size_t len);
ssize_t write(int fd, char buf[], size_t len);
```

copy.c

only I have permission!

```
int main (int argc, char * argv[]) {
    int fdin = open (argv[1], O_RDONLY);
    int fdout = open (argv[2], O_WRONLY | O_CREAT | O_EXCL, 0600);

    char buffer[1024];
    while (true) {
        ssize_t numRead = read (fdin, buffer, 1024);
        // should be error checking here
        if (numRead == 0) break;
        size_t numWritten = 0;
        while (numWritten < numRead) {
            numWritten += write (fdout, buffer + numWritten, numRead - numWritten);
        }
    }
    fclose (fdin);
    fclose (fdout);
    return 0;
}
```

each process has a process ID, info stored in data structure called process control block
- exactly 1 open file table for all PCBs to reference
- file descriptor 0 references the same thing on all processes
- also 1 system-wide vnode table - all processes alias the same vnodes