Syscalls: functions that ask the OS to do something we couldn’t do ourselves
return -1 on error

I/O calls:
* int open (const char* filename, int flags, ...)
  - G session ID
  - “file descriptor”
  - Info about what you want to do
  size_t read (int fd, void* dest, off_t size, count);
  - G how many bytes were read into buf
  size_t write (int fd, void* src, off_t size, count);
  - not guaranteed to write all `count` bytes
not in this class: int lseek (int fd, ...)
* int close (int fd);

Open flags:
- O_RDONLY: read only
- O_WRONLY: write only
- O_CREAT: create file if it doesn’t exist
- O_EXCL: raise an error if file already exists
File permissions

Read, write, execute

\[
\begin{array}{c|c|c}
\text{RUX} & \text{RUX} & 4 = R \\
110 & 101 & \text{(4+2)} \\
021 & & 6 \rightarrow RW \\
\end{array}
\] 

\[
\begin{array}{c|c}
\text{RUX} & 7 \rightarrow RWX \\
101 & \text{(2+1)} \\
\end{array}
\]

\[
\begin{array}{c|c}
3 \rightarrow \text{owner, group, everyone else} & 3 \rightarrow \text{WX} \\
\end{array}
\]

0644 \rightarrow \text{everyone else can R}

0640

What if we want RW for user, R for group, no access for everyone else?

Error handling

\[
\text{syscalls return -1 if something bad happens}
\]

\[
\text{Set errno to some number indicating cause of failure}
\]

\[
\text{ferror (const char * message): look at errno and print a human readable string}
\]
Key points for syscalls:
- Syscalls are how you interact with the "outside world"
- `open` is like `malloc`, `close` is like `free`
- Any time you call a syscall, check for errors
File sessions
program
open("hi.txt", ...) \to S
read(3, ...);

program?:
open("2.txt", ...) \to 3

Kernel: $\text{S} \leq \text{S}$

\{ 5:
  - "hi.txt"
  - cache.mode info
  - cursor: 0
3:
  - "2.txt"
  - mode info
  - cursor
4:
  - "hi.txt"
  - cursor: 0
?? - cache mode info