OS Implementation Has Evolved Over Time

Originally,
- the **array** of I/O channels in the OS
- **had fixed size**.

Today, most **OS's grow** the array **dynamically**.

But the **mechanism on the user side (programs) is the same**:
- an I/O channel is a small integer
- called a file descriptor.

Most I/O in C Uses Streams

In **C**,
- **most I/O uses an additional abstraction**
- built on top of file descriptors.

Streams provide
- a **continuous sequence of bytes**
- typically including some kind of **buffering**.

Buffering Happens for Both Reading and Writing

Buffering means
- **waiting until** a certain **amount or type of data is available** before sending anything, or
- **reading extra data** in anticipation of future requests for data.

For example,
- **when you type** at the keyboard
- **data** are **not** usually **delivered** to a program
- **until you press <Enter>**.
- That way, programs do not need to implement <Backspace>.

OS and Stream Both Buffer on Reads

Examples of buffering when reading…

Data on disk come in 4kB or 8kB blocks,
- and access time can be ~**10 msec**,
- so the **OS does not read 1B** from a file
- even if your program requests **1B**.

Similarly, making a **system call**
- is **too expensive** just **to obtain 1B**, so
- **streams read more and buffer** the rest
- **until your program asks for more**.