Buffered & Unbuffered IO

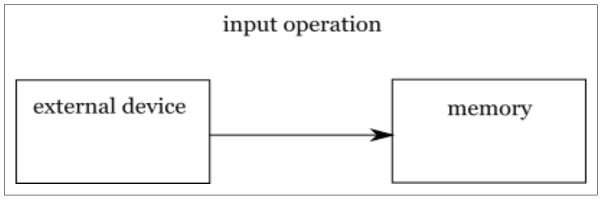
Input/Output (I/O)

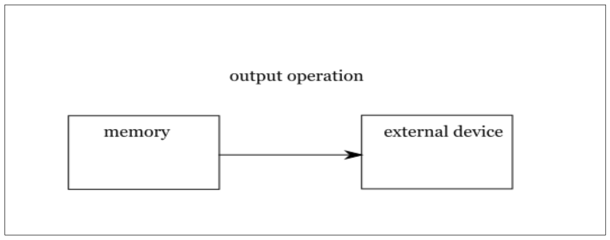
What is Input/Output

Input/Output (I/O) is the process of copying data between memory and external devices such as disk drivers, termials and networks.

Input operation copies data from an external device to memory.

Output operation copies data from memory to external device.



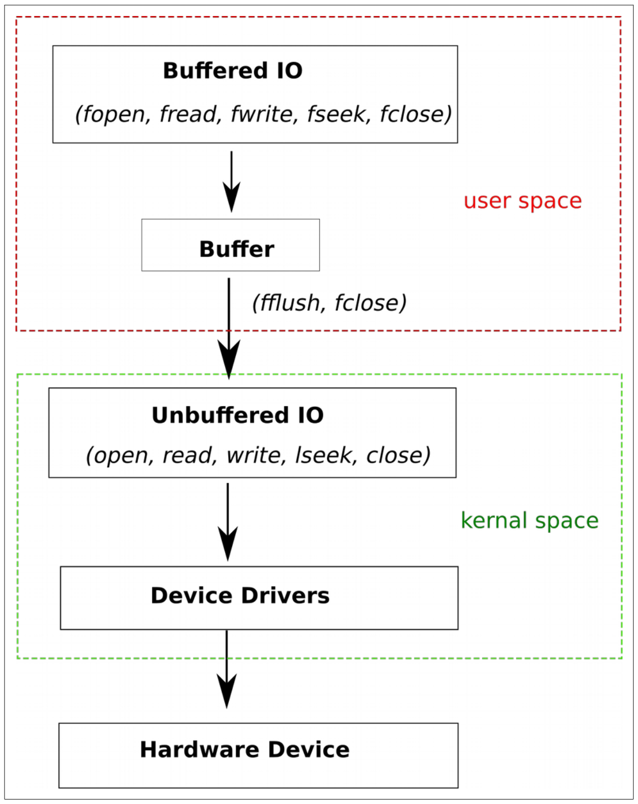


Buffered & Unbuffered IO

Unbuffered IO

Unbuffered IO is a part of POSIX, includes: open(), read(), write(), lseek(), close(), etc.

Unbuffered IO functions transfer data directly between memory and hardware device.



Buffered IO

Buffered IO is a part of ISO C, includes: fopen(), fread(), fwrite(), fseek(), fclose(), etc.

Buffered IO functions transfer data into a cache, that is called buffer.

The buffer will be written to hardware device in the following circumstances:

- The buffer is full.
- Function fclose() or fflush() is called.
- File is closed when process is terminated.

Buffered IO API is implemented by unbuffered IO API, in low level.

Buffered vs Unbuffered IO

Why are there two APIs for the same work?

They do the same work, but they give the different performances.

Unbuffered IO functions are expensive in each call.

Buffered IO functions is designed to reduce the call times.

Why Bubuffer IO functions are expensive?

There are two main reason.

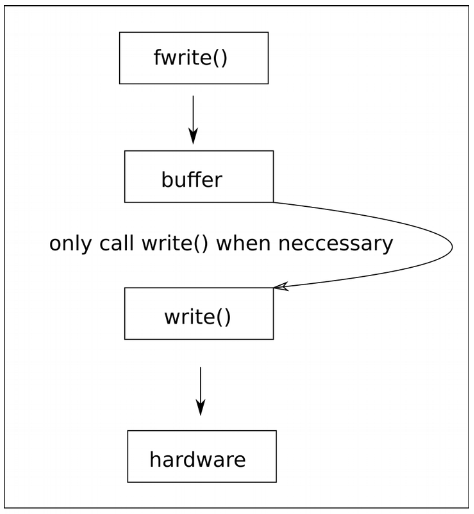
- Unbuffer IO functions like *open(), read(), write(), lseek(), close()* are also called system call (syscalls). Each time a syscall is invoked, the transitions between user space and kernal space are performed, such as copying data from user space to kernal space and back again. So, syscall is more expensive than a function call.
- The hardware has limitations and impose restrictions on the size of data blocks that can be read/written at a time. For example, the size of data block is 512 bytes; although we attempt to write 32 bytes, the driver still performs on 512 bytes.

How Buffered IO API give us better performance?

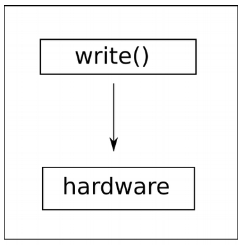
The key is the buffer.

With Unbuffer IO, the driver performs write operation onto hardware, each time *write()* is called.

With Buffer IO function *fwrite()*, data is cached into buffer. Syscall and write operation is executed only when necessary.



buffered io



Unbuffer IO