STREAM :

It is a full-duplex communication channel between a user-level process and a device in Unix System V and beyond

A STREAM consists of:

a) STREAM head interfaces with the user process

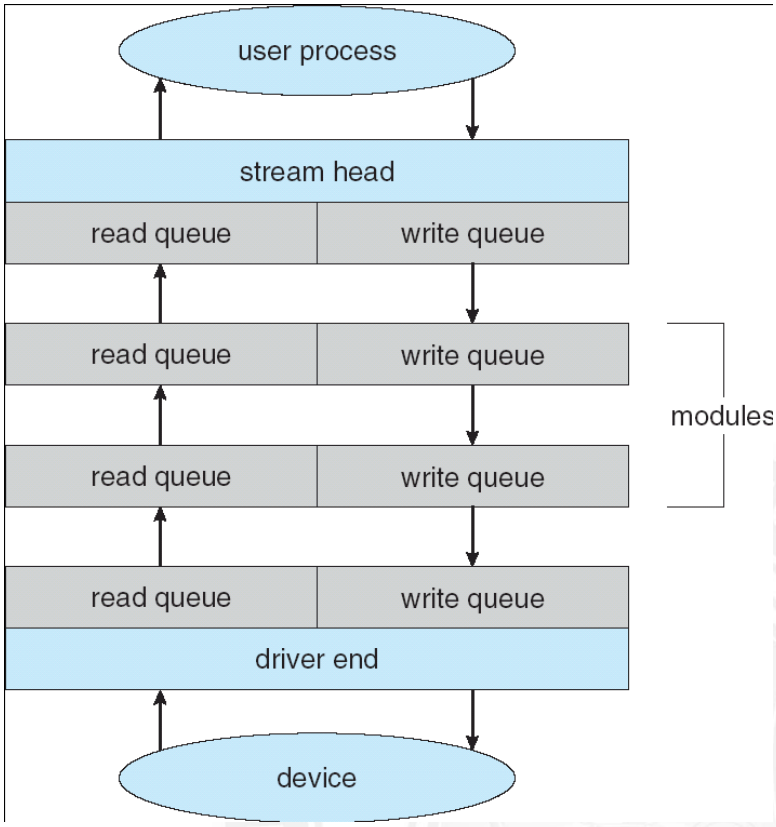
b) Driver end interfaces with the device

c) Zero or more STREAM modules between them.

Steps:

- The user process interacts with the ***stream head.***
- The device driver interacts with the ***device end.***
- Each module contains a read queue and a write queue
- Message passing is used to communicate between queues
- Modules provide the functionality of STREAMS processing and they are pushed onto a stream using the ioctl () system call.
- User processes communicate with the stream head using either read( ) and write( ) (or putmsg( ) and getmsg( ) for message passing.

The device driver **must** respond to interrupts from its device - If the adjacent module is not prepared to accept data and the device driver's buffers are all full, then data is typically dropped. Streams are widely used in UNIX, and are the preferred approach for device drivers. For example, UNIX implements sockets using streams.



Performance:

- I/O is a major factor in system performance:

We can employ several principles to improve the efficiency of I/O:

1. Reduce the number of context switches.

2. Reduce the number of times that data must be copied in memory while passing between device and application.

3. Reduce the frequency of interrupts .

4. Increase concurrency by using DMA-knowledgeable controllers .

5. Balance CPU, memory subsystem, bus, and I/O performance, because an overload in any one area will cause idleness in others.

Kernel I/O Subsystem

Kernels provide many services related to I/O.

- One way that the I/O subsystem improves the efficiency of the computer is by scheduling I/O operations.

- Another way is by using storage space in main memory or on disk, via techniques called buffering, caching, and spooling.

Services include:

1. I/O Scheduling:

To determine a good order in which to execute the set of I/O requests.

Uses:

a) It can improve overall system performance,

b) It can share device access fairly among processes, and

c) It can reduce the average waiting time for 1/0 to complete.

Implementation: OS developers implement scheduling by maintaining a —queue of requests‖ for each device.

1. When an application issues a blocking I/O system call,
2. The request is placed on the queue for that device.
3. The I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by applications.

2. Buffering:

Buffer: A memory area that stores data while they are transferred between two devices or between a device and an application.

Reasons for buffering:

a) To cope with a speed mismatch between the producer and consumer of a data stream.

b) To adapt between devices that have different data-transfer sizes.

c) To support copy semantics for application I/O.

Copy semantics: Suppose that an application has a buffer of data that it wishes to write to disk. It calls the write () system call.After the system call returns, what happens if the application changes the contents of the buffer?

With copy semantics, the version of the data written to disk is guaranteed to be the version at the time of the application system call, independent of any subsequent changes in the application's buffer.

A simple way that the operating system can guarantee copy semantics is for the write() system call to copy the application data into a kernel buffer before returning control to the application. The disk write is performed from the kernel buffer, so that subsequent changes to the application buffer have no effect.

3. Caching

A cache is a region of fast memory that holds copies of data.

Access to the cached copy is more efficient than access to the original

Cache vs buffer: A buffer may hold the only existing copy of a data item, whereas a cache just holds a copy on faster storage of an item that resides elsewhere.

When the kernel receives a file I/O request,

1. The kernel first accesses the buffer cache to see whether that region of the file is already available in main memory.
2. If so, a physical disk I/O can be avoided or deferred. Also, disk writes are accumulated in the buffer cache for several seconds, so that large transfers are gathered to allow efficient write schedules.

4. Spooling and Device Reservation:

Spool: A buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams.

A printer can serve only one job at a time, several applications may wish to print their output concurrently, without having their output mixed together

The os provides a control interface that enables users and system administrators ;

a) To display the queue,

b) To remove unwanted jobs before those jobs print,

c) To suspend printing while the printer is serviced, and so on.

Device reservation - provides exclusive access to a device

- System calls for allocation and de-allocation
- Watch out for deadlock

5. Error Handling:

- An operating system that uses protected memory can guard against many kinds of hardware and application errors.
- OS can recover from disk read, device unavailable, transient write failures
- Most return an error number or code when I/O request fails
- System error logs hold problem reports