5 Buffer Management

Content

- A journey of a byte
- Buffer Management

A journey of a byte

- Suppose in our program we wrote:

```
outfile << c;
```

- This causes a call to the **file manager** (a part of O.S. responsible for I/O operations)
- The O/S (File manager) makes sure that the byte is written to the disk.
- Pieces of software/hardware involved in I/O:
  - Application Program
  - Operating System/ file manager
  - I/O Processor
  - Disk Controller

- **Application program**
  - Requests the I/O operation
- **Operating system / file manager**
  - Keeps tables for all opened files
  - Brings appropriate sector to buffer.
  - Writes byte to buffer
  - Gives instruction to I/O processor to write data from this buffer into correct place in disk.
  - Note: the buffer is an exact image of a cluster in disk.
- **I/O Processor**
  - a separate chip; runs independently of CPU
  - Find a time when drive is available to receive data and put data in proper format for the disk
  - Sends data to disk controller
- **Disk controller**
  - A separate chip; instructs the drive to move R/W head
  - Sends the byte to the surface when the proper sector comes under R/W head.

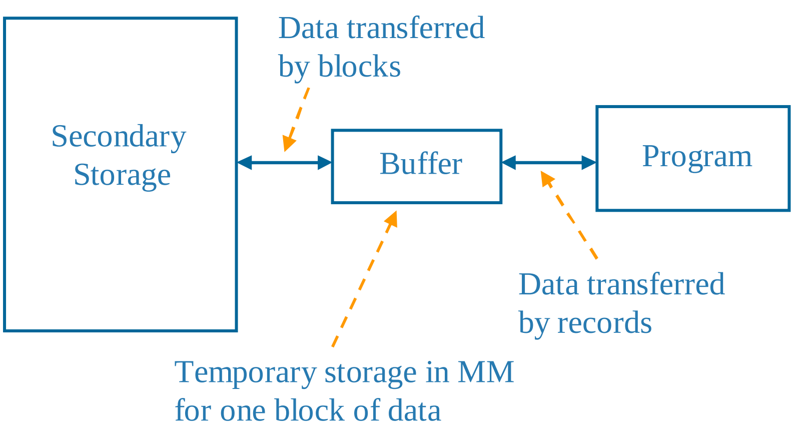
5 Buffer Management

Buffer Management

- Buffering means working with large chunks of data in main memory so the number of accesses to secondary storage is reduced.
- Today, we'll discuss the System I/O buffers. These are beyond the control of application programs and are manipulated by the O.S.
- Note that the application program may implement its own "buffer" – i.e. a place in memory (variable, object) that accumulates large chunks of data to be later written to disk as a chunk.

5 Buffer Management

System I/O Buffer



Buffer Bottlenecks

- Consider the following program segment:

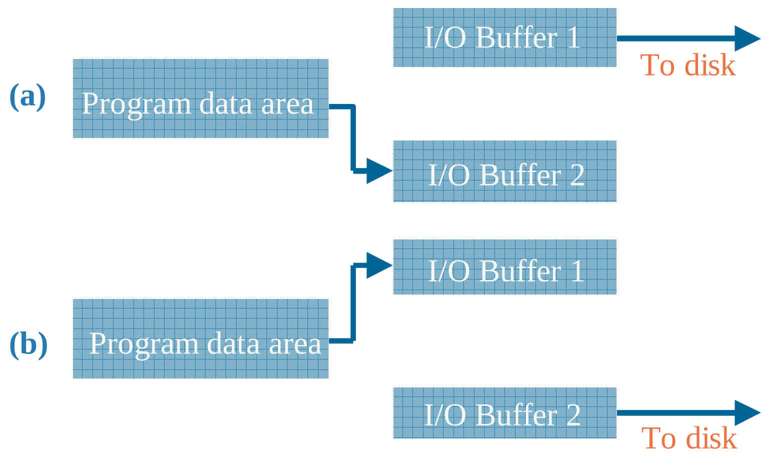
```
while (1) {
  infile >> ch;
   if (infile.fail()) break;
   outfile << ch;
}
```

- What happens if the O.S. used only one I/O buffer?
  ⇒ Buffer bottleneck
- Most O.S. have an input buffer and an output buffer.

Buffering Strategies

- **Double Buffering:** Two buffers can be used to allow processing and I/O to overlap.
  - Suppose that a program is only writing to a disk.
  - CPU wants to fill a buffer at the same time that I/O is being performed.
  - If two buffers are used and I/O-CPU overlapping is permitted, CPU can be filling one buffer while the other buffer is being transmitted to disk.
  - When both tasks are finished, the roles of the buffers can be exchanged.
- The actual management is done by the O.S.

Double Buffering



Other Buffering Strategies

- Multiple Buffering: instead of two buffers any number of buffers can be used to allow processing and I/O to overlap.
- Buffer pooling:
  - There is a pool of buffers.
  - When a request for a sector is received, O.S. first looks to see that sector is in some buffer.
  - If not there, it brings the sector to some free buffer. If no free buffer exists, it must choose an occupied buffer. (usually LRU strategy is used)

Buffer Management 5

Buffering Strategies: Move & Locate mode

- ▶ Move mode (using both system buffer & program buffer)
  - – moving data from one place in RAM to another before they can be accessed
  - – sometimes, unnecessary data moves
- ▶ Locate mode (using system buffer only or program buffer only)
  - – perform I/O directly between secondary storage and program buffer (program's data area)
  - – system buffers handle all I/Os, but program uses *locations* through pointer variable

Buffer Management 5

Move Mode and Location Mode