section10

Sunday, July 14, 2019 4:36 PM

section10

Section 10: Intro to I/O, Device Drivers, File Systems, FAT, and Queuing Theory

October 30, 2018

Contents

1 Warmup **2**
1.1 Short questions 2

2 Vocabulary **4**

3 Problems **6**
3.1 Disabling Interrupts 6
3.2 Disks 6
3.3 FAT 7
3.4 Queuing Theory 8

1 Warmup

1.1 Short questions

1. (True/False) If a particular IO device implements a blocking interface, then you will need multiple threads to have concurrent operations which use that device.

   True. A blockign thread would clo gup the interface.
   A nonblockign thread would get a rejection notice, allowing it to decide what to do next.
   Meanwhile, another thread can use the interface. This enables concurrency

2. (True/False) For IO devices which receive new data very frequently, it is more efficient to interrupt the CPU than to have the CPU poll the device.

   False. Interrupt / event processing is better for infrequent and unpredictable data streams.

3. (True/False) With SSDs, writing data is straightforward and fast, whereas reading data is complex and slow.

   False. Opposite. Writing data in SSDs is slower and more complicated. Erasing / wear levelling
   Becomes an important issue. Minimum size to write into a SSD is larger than disk

4. (True/False) User applications have to deal with the notion of file blocks, whereas operating systems deal with the finer grained notion of disk sectors.

   Abstractions are offset by 1 w.r.t. where they exist.
   Applications deal with files, OS deals with file blocks/

5. How does the OS catch a null pointer exception? Trace every action that happens in the OS based on what you have learned in cs162 so far.

   Hardware looks for VPN or address (0) and gets a TLB miss. Page table is looked up
   And, uh oh, VPN of 0 is marked invalid. OS proceeds to send a signal
   to the process (typically kill it)

6. What is a block device? What is a character device? Why might one interface be more appropriate than the other?

   Block device is a device that manages access of blocks of data (open, read, write, seek)

   Character device is a device that transfers data a single character at atime (keyboard, mouse)

7. Explain what is meant by "top half" and "bottom half" in the context of device drivers.

Top half is methods for kernel to invoke
Bottom half is a bunch of interrupt handlers

2 Vocabulary

- **I/O** In the context of operating systems, input/output (I/O) consists of the processes by which the operating system receives and transmits data to connected devices.
- **Controller** The operating system performs the actual I/O operations by communicating with a device controller, which contains addressable memory and registers for communicating with the CPU, and an interface for communicating with the underlying hardware. Communication may be done via programmed I/O, transferring data through registers, or Direct Memory Access, which allows the controller to write directly to memory.
- **Interrupt** One method of notifying the operating system of a pending I/O operation is to send a interrupt, causing an interrupt handler for that event to be run. This requires a lot of overhead, but is suitable for handling sporadic, infrequent events.
- **Polling** Another method of notifying the operating system of a pending I/O operating is simply to have the operating system check regularly if there are any input events. This requires less overhead, and is suitable for regular events, such as mouse input.
- **Response Time** Response time measures the time between a requested I/O operating and its completion, and is an important metric for determining the performance of an I/O device.
- **Throughput** Another important metric is throughput, which measures the rate at which operations are performed over time.
- **Asynchronous I/O** For I/O operations, we can have the requesting process sleep until the operation is complete, or have the call return immediately and have the process continue execution and later notify the process when the operation is complete.
- **Simple File System** - The disk is treated as a big array. At the beginning of the disk is the Table of Content (TOC) field, followed by data field. Files are stored in data field contiguously, but there can be unused space between files. In the TOC field, there are limited chunks of file description entries, with each entry describing the name, start location and size of a file.

  Pros and Cons

  The main advantage of this implementation is simplicity. Whenever there is a new file created, a continuous space on disk is allocated for that file, which makes I/O (read and write) operations much faster.

  However, this implementation also has many disadvantages. First of all, it has external fragmentation problem. Because only continuous space can be utilized, it may come to the situation that there is enough free space in sum, but none of the continuous space is large enough to hold the whole file. Second, once a file is created, it cannot be easily extended because the space after this file may already be occupied by another file. Third, there is no hierarchy of directories and no notion of file type.
- **External Fragmentation** - External fragmentation is the phenomenon in which free storage becomes divided into many small pieces over time. It occurs when an application allocates and deallocates regions of storage of varying sizes, and the allocation algorithm responds by leaving the allocated and deallocated regions interspersed. The result is that although free storage is available, it is effectively unusable because it is divided into pieces that are too small to satisfy the demands of the application.
- **Internal Fragmentation** - Internal fragmentation is the space wasted inside of allocated memory blocks because of the restriction on the minimum allowed size of allocated blocks.

- **FAT** - In FAT, the disk space is still viewed as an array. The very first field of the disk is the boot sector, which contains essential information to boot the computer. A super block, which is fixed sized and contains the metadata of the file system, sits just after the boot sector. It is immediately followed by a **file allocation table** (FAT). The last section of the disk space is the data section, consisting of small blocks with size of 4 KiB.

  In FAT, a file is viewed as a linked list of data blocks. Instead of having a "next block pointer" in each data block to make up the linked list, FAT stores these pointers in the entries of the file allocation table, so that the data blocks can contain 100% data. There is a 1-to-1 correspondence between FAT entries and data blocks. Each FAT entry stores a data block index. Their meaning is interpreted as:

  If $N > 0$, N is the index of next block

  If $N = 0$, it means that this is the end of a file

  If $N = -1$, it means this block is free

  Thus, a file can be stored in a non-continuous pattern in FAT. The maximum internal fragmentation equals to 4095 bytes (4K bytes - 1 byte).

  Directory in the FAT is a file that contains directory entries. The format of directory entries look as follows:

  Name — Attributes — Index of 1st block — Size

  Pros and Cons

  Now we have a review of the pros and cons about FAT. Readers will find most of the following features have been already talked about above. So we only give a very simple list of these features.

  Pros: no external fragmentation, can grow file size, has hierarchy of directories

  Cons: no pre-allocation, disk space allocation is not contiguous (accordingly read and write operations will slow), assume File Allocation Table fits in RAM. Otherwise lseek and extending a file would take intolerably long time due to frequent memory operation.

- **Queuing Theory** Here are some useful symbols: (both the symbols used in lecture and in the book are listed)
  - μ is the average service rate (jobs per second)
  - T_{ser} or S is the average service time, so $T_{ser} = \frac{1}{\mu}$
  - λ is the average arrival rate (jobs per second)
  - U or u or ρ is the utilization (fraction from 0 to 1), so $U = \frac{\lambda}{\mu} = \lambda S$
  - T_q or W is the average queuing time (aka waiting time) which is how much time a task needs to wait before getting serviced (it does not include the time needed to actually perform the task)
  - T_{sys} or R is the response time, and it's equal to $T_q + T_{ser}$ or $W + S$
  - L_q or Q is the average length of the queue, and it's equal to λT_q (this is Little's law)

Excess jobs arrived
Will be dropped

Average Queue Length / Average Queued Time = Average Arrival Rate

3 Problems

3.1 Disabling Interrupts

We looked at disabling CPU interrupts as a simple way to create a critical section in the kernel. Name a drawback of this approach when it comes to I/O devices.

We might miss important events, especially if there are many of them that overflow our buffer

3.2 Disks

What are the major components of disk latency? Explain each one.

Queue time: waiting in OS queue (disk is a finite resource that may have backlog)
Controller time: OS - DC communication
Seek time: translate the head to the right track
Rotational delay time: wait for the track to spin into the right position (beginning of a sector)
Read time: read data from your sector

In class we said that the operating system deals with bad or corrupted sectors. Some disk controllers magically hide failing sectors and re-map to back-up locations on disk when a sector fails.

If you had to choose where to lay out these back-up sectors on disk - where would you put them? Why?

Assuming failures are uniformly distributed, we should uniformly distribute our backups.

How do you think that the disk controller can check whether a sector has gone bad?

For some reasonable size of data, generate a checksum for it

Can you think of any drawbacks of hiding errors like this from the operating system?

Sector failures precipitate disk failures. OS / user might find this information useful.

3.3 FAT

What does it mean to format a FAT file system? Approximately how many bytes of data need to be written in order to format a 2GiB flash drive (with 4KiB blocks and a FAT entry size of 4 bytes) using the FAT file system?

2 E9 / (4E3) * 4 = 16,000 2 MB

Your friend (who has never taken an Operating Systems class) wants to format their external hard drive with the FAT32 file system. The external hard drive will be used to share home videos with your friend's family. Give one reason why FAT32 might be the right choice. Then, give one reason why your friend should consider other options.

FAT32 is supported by many OS

FAT32 has a max file size of 4gb

Explain how an operating system reads a file like "`D:\My Files\Video.mp4`" from a FAT volume (from a software point of view).

FAT volume is mounted in OS to path D:\
The first data block on the FAT volume has root directory, we search and open My Files
It then searches the subdirectory for Video.mp4
Now we have a poitner to the linked list that has the data blocks :)

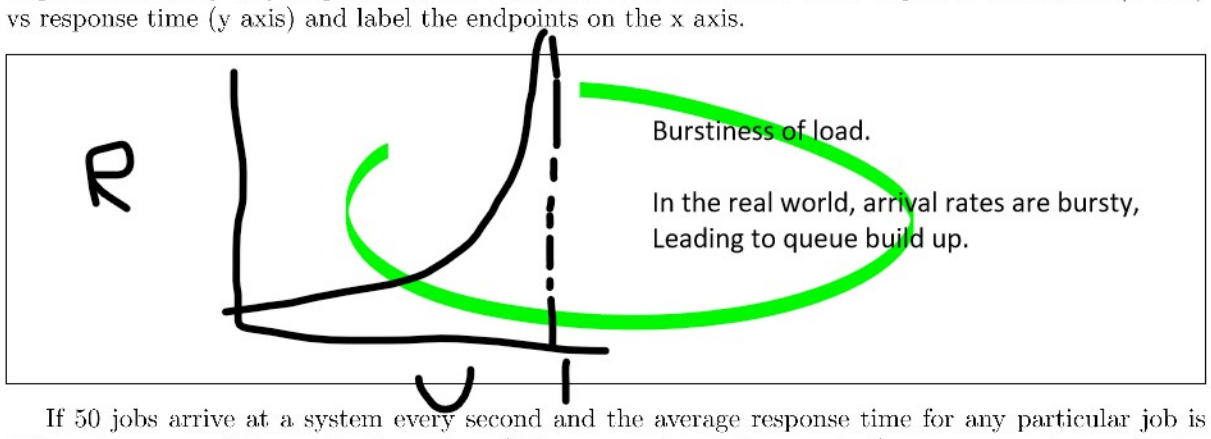
Compare bitmap-based allocation of blocks on disk with a free block list.

Bitmaps have fixed size metadata.

Free block lists are very low overhead, although you pay a word for every extension of a block, so that will start to add up. Hard to ma contiguous blocks here.

3.4 Queuing Theory

Explain intuitively why response time is nonlinear with utilization. Draw a plot of utilization (x axis) vs response time (y axis) and label the endpoints on the x axis.



If 50 jobs arrive at a system every second and the average response time for any particular job is 100ms, how many jobs are in the system (either queued or being serviced) on average at a particular moment? Which law describes this relationship?

Little's Law

Average Jobs in System = Average Arrival Rate * Average Job Time = 50* 0.1 = 5 jobs.

Is it better to have N queues, each of which is serviced at the rate of 1 job per second, or 1 queue that is serviced at the rate of N jobs per second? Give reasons to justify your answer.

1 queue with N JPS.

With the same load, an N queue system might have uneven load balancing, potentially having One queue backed up while other queues are idle.

What is the average queueing time for a work queue with 1 server, average arrival rate of λ, average service time S, and squared coefficient of variation of service time $\mathbf{C}$?

Tq = Tser (\mu / (1-\mu)) * (C+1) / 2

\mu := \lambda * S

What does it mean if $\mathbf{C} = 0$? What does it mean if $\mathbf{C} = 1$?

CoV = 0 implies a constant arrival rate.

C = 1 means a stochastic arrival rate, that can be modeled as a poisson distribution