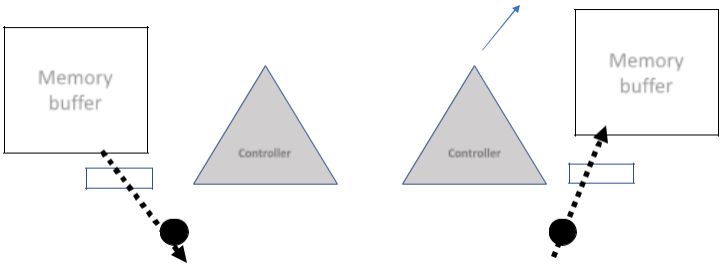
Romil V. Shah
20008692

CS 520
Prof. Igor Faynberg

Homework Assignment #7
Submission Date: 12/06/2022

Q1.

A local area network is used as follows. The user issues a system call to write a data packet to the network. The operating system then copies these data to a kernel buffer. Then it copies these data to the network controller board. When all the bytes are safely inside the controller, they are sent over the network at a rate of 10 megabits/sec. The receiving network controller stores each bit a microsecond after it is sent. When the last bit arrives, the destination CPU is interrupted, and the kernel copies the newly arrived packet to a kernel buffer to inspect it. Once it has figured out which user the packet is for, the kernel copies the data to the user space. (This is illustrated by the figure below).



If we assume that each interrupt and its associated processing takes 1 millisecond (msec), that packets are 1024 bytes long (including the headers), and that copying a byte takes 1 microsecond (μsec), what is the maximum rate at which one process can pump data to another? Assume that the sender is blocked until the work is finished at the receiving side and an acknowledgement comes back. For simplicity, assume that the time to get the acknowledgement back is so small that it can be ignored. [Suggestions: Observe 1) how many times a packet is copied and how long it takes, 2) how many interrupts are there to be processed and how long that takes, and 3) how long the transmission of a packet takes. Then calculate the overall data transmission rate.]

Sol.

We are given the following input:
Interrupt Process Time = 1 msec i.e., 1000 micro sec
Packet Size = 1024 bytes (including header)
Time to copy a byte = 1 micro sec

We need to find the maximum data rate. For that, first we find the no. of time the packet is copied and then, we find the no. of interrupts that are needed to be executed.

In this process, a packet is to be copied for 4 times and time for this operation is 4.1 msec. Also, there are 2 interrupts concerned in this process. Each of them takes 1 msec. So, 4.1 msec + 2 msec = 6.1 msec. The transmission time is 0.83 msec for the total of 6.93 msec per packet.

So, the overall data transmission rate is 147,763 bytes per sec.

Q2.

A computer can read or write a memory word in 10 nsec. The computer has 34 CPU registers (including the program counter and status word), each one word long. When an interrupt occurs, all these registers are pushed onto the stack. What is the maximum number of interrupts this machine can handle in a second?

Sol.

1 nsec = 10^{-9} sec.

We have been given the following,
Time to read / write a memory word = 10 nsec = 10^{-8} sec
Number of CPU registers = 34.

Each register is one word long, the number of words is 34.

When an interrupt occurs, all the registers are pushed on the stack and after that to continue the execution all of them are popped from the stack.

It means a total of 68 words are to be read/write.

Total time = 68 * 10^{-8} sec
So, one interrupt takes 68 * 10^{-8} sec

So, the total number of interrupts the computer can handle in one sec are
= 1 / (68 * 10^{-8})
= 1470588.2

Q3.

A floppy disc is doubly interleaved. It has eight sectors of 512 bytes per track, and a rotation rate of 300 rpm. How long does it take to read all the sectors of a track in order, assuming the arm is already correctly positioned, and half rotation is needed to get sector 0 under the head? What is the data rate? Now, solve the same problem for the case without interleaving. How much does the data rate degrade because of interleaving?

Sol.

Given that,

No. of sectors = 8
Rotation Rate = 300 Rotations Per Minute
Total bytes of 8 sectors = 512 * 8 = 4096 bytes.

Data Rate for Doubly Interleaved Disc:

Rotations to read a whole track in order from 0 to 7 = 0.375 * 7 = 2.625
(0.375 as the disc is double interleaved)

Total Rotation = 0.5 (Half rotation for sector 0) + 2.625 = 2.125 Rotations

Time for 3.125 rotation = (3.125 / 300) * 60 = 0.625 seconds

Data Rate = 4096 / 0.625 = 6553.6 bytes/sec

Now, calculating without the interleaving,
Data Rate for Disc with no interleaving:

Rotations to read a whole track in order from 0 to 7 = 0.125 * 7 = 0.875 Rotations

Total rotation = 0.5 (Half rotation for sector 0) + 0.875 = 1.375 Rotations

Time for 1.375 rotation = (1.375 / 300) * 60 = 0.275 seconds

Data Rate = 4096 / 0.275 = 14894.55 bytes/sec

Data Rate Degradation due to the interleaving = (6553.6 / 14894.54) * 100 = 44%

So, the data rate degrades about 44% because of interleaving of the disc.

Q4.

Disk requests come into the disk driver for cylinders 10, 22, 20, 2, 40, 6, and 38, in that order. A seek takes 6 msec per cylinder moved. The arm is initially at cylinder 20. How much seek time is needed if a) FCFS, b) SSTF, or c) SCAN (elevator) algorithm is employed?

Sol.

Order of disk requests: 10, 22, 20, 2, 40, 6, 38
Seek Time Per Cylinder Movement = 6 msec

a) FCFS:
    a. Total number of cylinders travelled by arm =
    = (20 – 10) + (22 – 10) + (22 – 20) + (20 – 2) + (40 – 2) + (40 – 6) + (38 – 6)
    = 10 + 12 + 2 +18 + 38 + 34 +32
    = 146

    b. Total seek time = 146 * 6 = 876 msec.

b) SSTF:
    a. Total number of cylinders travelled by arm =
    = (20 – 20) + (22 – 20) + (22 – 10) + (10 – 6) + (6 – 2) + (38 – 2) + (40 – 38)
    = 0 + 2 + 12 +4 + 4 + 36 +2
    = 60

    b. Total seek time = 60 * 6 = 360 msec.

c) SCAN:
    a. Total number of cylinders travelled by arm =
    = (20 – 10) + (22 – 20) + (38 – 22) + (40 – 38) + (40 – 10) + (10 – 6) + (6 – 2)
    = 10 + 2 + 16 + 2 + 30 + 4 + 4
    = 58

    b. Total seek time = 58 * 6 = 348 msec.

Q5.

The Stupido v1.0 operating system only supports a single directory but allows that directory to have arbitrarily many files with arbitrarily long name. Furthermore, every printable character is allowed as part of a file name. Can you suggest a naming convention that allows to simulate a hierarchical (i.e., nested directory) file system?

Sol.

We can create the file name in such a way to appear like the file path in the hierarchical file system.

An example of the file name: /user/documents/cs520/assignments/hw7/hw7.txt

Q6.

The i-node of Figure 1 contains 10 direct addresses of 4 bytes each and a pointer to a block of addresses. All addresses are four-bytes long; each block is 1024 bytes. What is the size of the largest file possible?

Sol.

No. of blocks = 10.
Size of an address = 4 bytes
Size of a block = 1024 bytes.
Total size of the 10 blocks already given = 1024 * 10 = 10240 bytes

Now, if the i^{th} node contains indirect address, then no. of blocks
= Size of a block / No. of Bytes of an address
= 1024 / 4
= 256 bytes.

256 block size = 256 * 1024 = 262144 bytes.

The size of the largest file which is possible is 262144 + 10240 = 272384 bytes.

Q7.

After the disk partition is first formatted, the beginning of a free space bitmap looks like this: 1000 0000 0000 0000 (the first block is used by the root directory). As we discussed during the lecture, the system always searches for free blocks starting at the lowest numbered block, so after writing file A, which uses six blocks, the bitmap would change to 1111 1110 0000 0000.... Show the bitmap after each of the following actions: a) File B is written, using five blocks; b) File A is deleted; c) File C is written, using eight blocks; d) File B is deleted.

Sol.

Start of a free space bitmap: 1000 0000 0000 0000

Bitmap after writing file A: 1111 1110 0000 0000

Bitmap after writing file B: 1111 1111 1111 0000

Bitmap after deleting file A: 1000 0001 1111 0000

Bitmap after writing file C: 1111 1111 1111 1100

Bitmap after deleting file B: 1111 1110 0000 1100

Q8.

What is more efficient way of caching file data—caching through physical disc blocks or caching using virtual memory? Explain your answer.

Sol.

Caching file data using virtual memory is more efficient and feasible.

Virtual memory is lightweight, feasible and efficient method for caching file data. It allows the operating system to intercept every page fault and determine if the file is the primary copy in memory or not.

If the file is primary, the operating system will keep it in memory until the application needs it. When the application needs the file, the OS will intercept the request and load the file from memory. If it is not primary, the operating system will load it from the disk into memory. Since all files are loaded at the same time, this can happen very quickly.

As a result, all page faults in the virtual memory system will always be satisfied.

While, in a physical memory cache, this might not be the case every time. The operating system must intercept each page fault and check to see if the data is in memory. If not, it must load the file data from the disk into memory. The operating system must always do this, even if the data is present in memory.

If the file data is cached in physical memory, it must intercept all page faults, check if the file is the primary copy in memory, and if not, load the file from the disk. This process is more expensive than the virtual memory cache.

Q9.

Explain why using a unified buffer cache results in better performance of a file system.

Sol.

A unified buffer cache uses the same, single buffer cache for caching pages for both memory-mapped IO as well as ordinary IO.

A unified buffer cache adds data to the cradle reserve. Instead of going to the disk every time the unified buffer cache lets different updates happen in memory. When a buffer has been in memory for a certain amount of time, the support flushing daemon (bdflush) sends it to the disk.

Memory use can have a big effect on how something is run. Cache memory stores frequently used instructions and data that the processor might need to access instantly. Its access is faster compared to RAM as it is on the same chip as the processor. This makes it less likely that the CPU might wait for slower memory to be recovered from primary memory repeatedly.

In this way, we can say that a file system runs better when a unified buffer cache is used.

Q10.

Explain how the data structures of the Linux file system serve to map file names to blocks of a storage device. (Use the man page descriptions to obtain the information you need for this and the next problems.)

Sol.

The files are stored on a disk by their index node and not by their name. The index node contains information about the actual data and its location on the hard disk. But, the index node does not contain any information regarding the file name. Therefore, directories help to map the file names to their index node numbers.