IIT Guwahati - Department of Computer Science & Engineering

CS343-Operating Systems- Quiz #4 [09.11.2021]

Q1: **Long answer [to type in text box], 2 marks.**

When DMA controller is working in burst mode the system bus is not under the control of CPU and hence CPU cannot access the contents of main memory. In this situation, does that mean program fetching and execution is stalled and CPU is idle? Explain your answer with necessary justifications.

Answer: System bus connects processor and memory. If DMA controller is using system bus in burst mode, CPU cannot connect to main memory. Hence, if program execution has to continue there should be some other memory from which program is fetched and executed in CPU. Cache memory/any other memory inside CPU chip that are connected to CPU helps in scenario. When DMA controller is working in burst mode the program fetching and execution is not stalled and CPU due to the presence of cache memory.

[2 marks for the right explanation, 1 mark if partially correct]

Q2: File Upload [only one page allowed. make sure the scan and the illustrations are clear, 3 marks]

Consider a file system F that has 256 blocks in total and follows a linked file allocation scheme. The file control block is always kept in main memory. In F, consider a file **f1** that has 128 blocks. Calculate how many disk reads and writes are required in total if we need to add a new block in the middle of **f1** followed by removing a block from the end of **f1**. Explain the steps involved in carrying out the above-mentioned operations.

Answer: The disk I/O operations of files happens based on request from main memory. Every file operation even though they happen in sequence is treated individually as file systems do not keep track of current location and disk operations are triggered by accessing respective file control blocks. For linked allocation policy, every file access needs to start from file control block and has to traverse from the beginning of the list.

Here f1 is stored in 128 blocks. Let bx be the new block to be written.

To write a new block in middle of f1, we need to do read b1 to b64, once b64 is read we get link to b65. The new block bx is written with this link stored in it. And finally, the b64 is written back with address of new block added to it. So, we have 64 reads and 2 writes. Now we have total 129 blocks after adding bx.

Now to remove the last block (129^{th} block), we need to read 128 blocks and then replace its next link with null. The 128^{th} block is written back with updated link. Hence, we have total of 128 reads and 1 write to facilitate the removal of last block. **[1 mark for explanation]**

Total number of read operations: 64 + 128 = 192 **[1 mark]**

Total number of write operations: 2+ 1 = 3 **[1 mark]**