Steven Tardo
4 May 2015
CSCI 4401
Homework 2

1) Mutual Exclusion holds true because only one car can be in one quadrant of the intersection at a time. Hold and wait occurs if each car pulls into the first quadrant they need but no car ever moves to give up the second quadrant needed by another. No preemption occurs because a car cannot forcibly make another car move from a quadrant (short of crashing into them). Circular wait occurs because each car if waiting for the next car in a circle, and if nobody ever moves the chain will never exit.

2) a) If foo executes semWait(S) and changes the value of S to 0, then bar executes semWait(R) and changes the value of R to 0, then when foo attempts to execute semWait(R) it will be blocked and when bar attempts to execute semWait(S) it will be blocked, resulting in both blocked forever.

   b) No, since concurrent execution could result in both of them blocked forever, if the processes are run in the correct order, they will both run eventually. If the execution sequence is incorrect then they will both be blocked.

3) a) If all of the philosophers at the table are lefties, if each picks up their left fork, then none of them can pick up their right fork. Therefore there needs to be at least one philosopher who is a righty in order to prevent deadlock

   b) If all of the philosophers at the tables are lefties, there is a chance that at least one of the philosophers will never get the chance to pick up the right fork and will starve. Therefore there needs to be at least one righty at the table so that one of the lefties does not starve.

4) a) The maximum size of the swapped out process would be 4M, since that is the biggest block with free space available that occurs before the block where the 2M was placed just before. The process could not have been swapped out after where the 2M was places, because the 2M would have ended up in the 4M spot at the beginning.

   b) The size of the free block just before X partitioned it was 7M. This is because X is 2M and the remaining free space is 5M. The 1M space just before the 2M Is not included in this block, because X would have taken up that spot if it was part of the block.

c) If the best-fit algorithm is used, the 3M process would be placed in the last free space available, because it is exactly 3M. If the first-fit algorithm is used then the 3M process would be placed in the first available space (4M) because it is large enough to hold it. If the next-fit algorithm is used, then the 3M process will be placed in the 5M space right after where the 2M was placed. If the worst-fit algorithm is used, then the 3M process will be placed in the 8M free space, because that would leave the most memory wasted.

5) a) Since there are 2^10 bytes in each page, and there are 2^16 pages, then each page's logical address would therefore need to be 26 bits.

b) Since memory is divided into frames of equal length and processes are divided into pages the same length as the frames, then there has to be 2^10 bytes in a frame.

c) Since there are 2^32 bytes of physical memory, and each page is 2^10 bytes, which is the same size as the frame, then 22 bits are required to specify the frame.

d) Since there are 2^16 pages of logical address space, and each page needs an entry in the page table, then there has to be 2^16 entries in the page table.

e) Since 22 bits are required to specify the frame, and 1 bit is needed for valid/invalid, 23 bits must be in each page table entry.

6) a) The physical address is going to be 660 + 198 which is 858. Since 198 is less than 248, a segment fault does not occur.

b) The physical address is going to be 222 + 156 which is 378. Since 156 is less than 198, a segment fault does not occur.

c) 530 is greater than 422 so a segment fault will occur.

d) The physical address is going to be 996 + 444 which is 1440. Since 444 is less than 604, a segment fault does not occur.

e) The physical address is going to be 660 + 222 which is 882. Since 222 is less than 248, a segment fault does not occur.

7) a) Since four operations take up one data page, and the result also needs to be stored, and there are three pages that can be used for data, 3 operations + 3 operations + 3 operations, or 9 operations, can occur before a page fault.

b) The program can be modified to minimize page fault frequency by overwriting either A or B with the answer, then moving them to C at the end.

c) The modification would allow for one more operation per page, resulting in 12 operations before a page fault.

8) a) (7), (7,0), (7,0,1), (2,0,1)F, (2,0,1), (2,3,1)F, (2,3,0)F, (4,3,0)F, (4,2,3)F, (4,2,3), (0,2,3)F, (0,2,3), (0,2,3)

b) (7), (7,0), (7,0,1), (2,0,1)F, (2,0,1), (2,0,3)F, (2,0,3), (4,0,3)F, (4,0,2)F, (4,3,2)F, (0,3,2)F, (0,3,2), (0,3,2)

c) (7*), (7*, 0*), (7*, 0*, 1*), (2*,0,1)F, (2*,0,1), (2*,3*,1)F, (2*,3*,0*)F, (4*,3,2)F, (4*,3,2), (4*,3,2), (4*,0*,2)F, (4*,0*,3*)F, (2*,0,3)F

d) (7), (7,0), (7,0,1), (2,0,1)F, (2,0,1), (2,0,3)F, (2,0,3), (2,4,3)F, (2,4,3), (2,4,3), (2,0,3)F, (2,0,3), (2,0,3)

e) a: 6, b: 6, c: 7, d: 4

9) a) If a memory reference takes 200 ns, then a paged memory reference must take 400 ns, since the page table is also in memory.

b) (220*0.85) for hit + (420*0.15) for a miss = 250 ns.

c) Since an extra 200 ns is required to reference the page table, it would be ideal to find the reference in the TLB as often as possible so the extra 200 ns is not wasted going to the page table.

10) For RAID 0, the amount of storage space would be 4*200 = 800 GB.
For RAID 1, the amount of storage space would be 4*200/2 = 400 GB.
For RAID 3, the amount of storage space would be 3*200 = 600 GB, since only one disk is needed for redundancy.
For RAID 4 and 5, the amount of storage space would also be 600 GB, the only thing that changes is how the data is arranged.
For RAID 6, since 2 more disks are needed than the amount of storage space, the amount of storage space would be 2 * 200 = 400 GB.

11) a) Indexed sequential would probably be best for this type of file organization. Indexed sequential file management allows for quick access to fields, and since data is updated infrequently, it doesn't matter what order in which it is placed.

b) Sequential would probably be best for this type of file organization. Sequential file management allows for optimization if applications require the processing of all the records in a file.

c) Direct or Hashed file management would probably be best for this type of file organization. This is because direct or hashed file management has no concept of sequential ordering, which means updating records will go quickly. Direct or hashed file management also allows for rapid access when required.

12) a) If each address is 32 bits, then each address is 4 bytes. If the block size is 8K, then each block can hold 2000 addresses. For direct, since there are 12, using Table 12.3 as a reference, the number of bytes is 12 * 8K, which is 96K. In single indirect, 2000 addresses * 8K = 16MB. In double indirect, 2000 addresses * 2000 addresses * 8K = 32 GB. In triple indirect, 2000 addresses * 2000 addresses * 2000 addresses * 8K = 64 TB. Therefore the maximum file size supported is over 64 TB.

b) Since 24 bits are required to represent a physical block, then there are 2^24 possible addresses to identify a block. Since each block size is 8K, then the maximum file system partition is 2^24 * 8000 = 134.218 gigabytes.

c) Since direct access can only access up to 96000 bytes, and single indirect can access up to 16000000 bytes, then it would take two disk access to access the byte in position 13,423,956.