CIS 520 - Operating Systems I – Homework #4

Due: Wednesday, Nov. 6th, by 11:59 pm, upload via K-State OnLine

1. Dynamic Memory Management: A dynamic memory allocator uses a linked list to track free blocks. Suppose that the free list contains just two blocks, of size 24 and 16 bytes, in that order. Ignore any space required for bookkeeping overhead.

   (a) List a sequence of malloc( ) calls that would succeed using first-fit allocation, but fail with best-fit, or explain why such a sequence cannot exist.

   (b) List a sequence of malloc( ) calls that would succeed using best-bit, but fail with first-fit, or explain why such a sequence cannot exist.

   (c) List a sequence of malloc( ) calls that would succeed on the original free list, with free blocks of size 24 and 16, using worst-fit, but fail with first-fit, or explain why such a sequence cannot exist.

   (d) In a pure paging system, processes are allocated a fixed number of pages. Each page has a fixed size. So, typically, half of the last page is wasted when memory is allocated. Is this an example of internal or external fragmentation? Explain briefly.

   (e) For dynamic memory management, if the process calls malloc( ) with a request for 14 bytes and the allocator is using the best-fit algorithm, the request for 14 bytes will be satisfied by breaking the block of size 16 into two parts, and returning the remaining 2 bytes to the free list. This small block of size 2 bytes is sometimes referred to as "sawdust" because it will probably never be used. Is this an example of internal or external fragmentation? Explain briefly.

2. Paging and Page Replacement Algorithms:

   (a) Discuss situations in which the most recently used (MRU) page replacement algorithm generates fewer page faults than the least recently used (LRU) page replacement algorithm. Also, discuss under what circumstances the opposite holds.

   (b) Discuss the philosophy behind the second chance (clock) algorithm and how it approximates LRU. Drawings are fine here to help with your explanation.

   (c) What is Belady's anomaly?

   (d) Can the FIFO algorithm exhibit Belady's anomaly?

   (e) Can the LRU algorithm exhibit Belady's anomaly?

3. System Calls and Swapping:

   (a) Your teammates suggests that, in order to check the validity of the buffer passed to the write( ) system call (which is declared as **write(int fd, const void *buffer, size_t length)**, you could simply check whether the beginning of the buffer ('**buffer**') and the end of the buffer (' **buffer+length–1**') are in the user virtual address range and have valid page table entries. Is this correct? Either briefly explain why this approach is correct or give an example user program for which this approach fails.

   (b) In Unix terminology, a child process becomes a 'zombie' if it calls exit() before its parent calls wait(). Your system administrator claims that too many zombie processes are bogging down the machine. Is he correct? Briefly justify your answer.

   (c) In Pintos, if the parent process calls wait() after the child has already called exit(), how can we ensure that the parent knows that the child process is a zombie (has already exited)? Likewise, how can we block the parent if the parent calls wait() before the child calls exit()?