CS 354 (Park) **Final** **Dec. 18 (Fri.), 2015**

Remarks: Keep the answers compact, yet precise and to-the-point. Long-winded answers that do not address the key points are of limited value. Binary answers that give little indication of understanding are no good either. Time is not meant to be plentiful. Make sure not to get bogged down on a single problem.

PROBLEM 1 (36 pts)

(a) What is the optimal offline page replacement strategy (i.e., Belady)? What is the LRU policy and how is it related to the optimal offline strategy? Why is LRU not used in practice? What is global clock and how is it related to LRU?

(b) How has the "mice and elephants" property of real-world file systems (i.e., most files are small, a minority are very large) influenced traditional file system design? Why are kernels able to handle read requests more efficiently than write requests during file I/O? Suppose for a disk-based file server with disk bandwidth B bytes per second that is busy serving client requests, it is found that only 15% of B is being utilized. Given that the server fields many requests, what may cause its disk bandwidth to be severely underutilized?

(c) The Solaris TS scheduler implemented in lab3 achieves constant (i.e., $O(1)$) overhead with respect to time complexity. Explain why this is the case by considering enqueue and dequeue operations separately. In lab2, a form of fair scheduling was implemented where the priority of a process was set to $1/$*CPU-time-received*. Thus a process that has received relatively little CPU time would get a higher priority than a process that has consumed more CPU time. Why is a multilevel feedback queue not extensible for fair scheduling? What is the scheduling overhead associated with fair scheduling?

PROBLEM 2 (36 pts)

(a) What is memory thrashing? What are its typical symptoms? Is memory thrashing possible if virtual memory is turned off (i.e., the default configuration of the XINU kernel we have been using)? Will increasing memory (i.e., RAM) help alleviate the problem? What about increasing CPU speed? Explain your reasoning.

(b) How is a tickless kernel different from a tickful kernel? What is its main strength? What is its main weakness? How might a hybrid design that aims to achieve the best of both worlds work?

(c) Explain why modern kernels divide the lower half of the kernel that responds to interrupts into two subsystems: a top half that directly interfaces with a device controller upon receiving an interrupt and a bottom half that processes data that the top half has copied into a kernel buffer. What are the two design choices for implementing the bottom half? What are their pros/cons? In today's computing systems with DMA support for multimedia streaming, how does the role of the top half change? What is the main performance benefit that DMA brings?

PROBLEM 3 (28 pts)

(a) The context switch cost increased significantly when we introduced virtual memory-based demand paging. What are the reasons for that? Despite the increased context switch cost and extra hardware expenditure incurred to build VM support, what benefits enabled by VM outweigh the negatives? Page faults in kernels with disk-based file systems induce a context switch because disks are significantly slower than main memory (i.e., RAM): a process that page faulted cannot continue execution until the missing page is brought in. Suppose in the future the speed difference between RAM and flash memory (i.e., SSD)—used increasingly in place of disks—narrows to the extent that there is only a negligible gap. How might page fault handling be done differently in such an environment?

(b) In XINU, we encountered the delta list data structure when discussing the implementation of the sleep queue. What is the main benefit of using a delta list for the sleep queue? In the XINU kernel, the time slice of the current process was updated by making the clock interrupt handler decrement it by 1 msec each time it was invoked. (XINU is a tickful kernel.) By generalizing the sleep queue into an event queue that is implemented as a delta list, we may use the event queue to manage both sleep and time slice depletion events. Describe how you would go about modifying XINU to implement a unified event queue that handles both types of events. As a further step, how would you modify XINU to turn it into a tickless kernel that makes use of the unified event queue and the x86 interval timer (i.e., PIT)?

BONUS PROBLEM (10 pts)

Why is isolation/protection an essential feature of general-purpose computing systems? How did isolation/protection influence the design and implementation of asynchronous IPC with callback function in lab4?