FOR EXTERNAL EXAMINER (date of this version: 16/3/2017)

UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR09047 OPERATING SYSTEMS

Friday 12th May 2017

14:30 to 16:30

INSTRUCTIONS TO CANDIDATES

Answer any TWO of the three questions. If more than two questions are answered, marks from QUESTION 1 and QUESTION 2 on the exam paper will be taken.

All questions carry equal weight.

CALCULATORS MAY NOT BE USED IN THIS EXAMINATION

Year 3 Courses

Convener: C. Stirling
External Examiners: A. Cohn, A. Donaldson, S. Kalvala

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. (a) Briefly explain what a race condition is. *[2 marks]*

   (b) Using the following hardware supported, atomic test and set() function:

```
bool test_and_set(bool *flag) {
  bool old = *flag;
  *flag = True;
  return old;
}
```

   i. Write 2 functions BusyAquire() and BusyRelease() in C or pseudo-code which implement simple spin locks. *[2 marks]*

   ii. Write 2 new functions NonBusyAquire() and NonBusyRelease() which avoid busy-waiting. You are provided with a wait queue and two functions Enqueue_ process(), which puts a calling process into the wait queue and puts it to sleep and Dequeue_process() which removes one waiting process from the wait queue and places it at the front of the ready queue. *[6 marks]*

   (c) Explain deadlock prevention, deadlock avoidance and deadlock detection, highlighting any differences. *[4 marks]*

   (d) Five philosophers sit at a round table with bowls of spaghetti. A fork is placed between each pair of adjacent philosophers, 5 forks in total. Each philosopher alternately thinks and eat. However, a philosopher can only eat spaghetti when he has both left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it is not being used by another philosopher. After he finishes eating, he puts down both forks so they become available to others. A philosopher can take the fork on his right or the one on his left, one at a time, as they become available, but cannot start eating before having both of them.

   Depending on how the philosophers pick up forks, they may deadlock, eternally waiting for each other to release a fork.

   i. Provide one example of when such a deadlock can occur. *[2 marks]*

   ii. Show how this deadlock problem can be fixed using:

      A. Deadlock prevention *[2 marks]*

      B. Deadlock Avoidance using Bankers algorithm *[3 marks]*

      C. Deadlock detection *[2 marks]*

   iii. In your opinion, what is the best method and why? *[2 marks]*

2. (a) Draw a diagram highlighting how the MMU uses limit and base registers to translate logical addresses into physical addresses. Why is this necessary? *[3 marks]*

   (b) Explain how fragmentation takes place in fixed and variable partition allocation. *[3 marks]*

   (c) This question concerns segmented memory. Draw a diagram explaining how logical segmented memory addresses are translated to physical memory. How does segmented memory tackle memory fragmentation? *[4 marks]*

   (d) Paging is now used on modern systems to prevent fragmentation.

      i. Draw a labelled diagram explaining how a logical address is translated into a physical address with paging hardware. You should include the TLB and page table in your diagram. *[4 marks]*

      ii. In a 64 bit machine with 4KB pages, how large is the page table? *[2 marks]*

      iii. Explain one method to reduce this size *[3 marks]*

   (e) Virtual memory is a mechanism that supports larger logical memory than physical memory by using physical memory as a page cache to disk. Consider a system with only 3 frames i.e. only 3 pages can sit in memory with all pages initially on disk. Given the following page reference sequence 1,2,3,4,1,2,5,1,2,3,4,5 calculate the number of page faults when using the following replacement policies:

      i. Not recently used, 2nd chance clock *[2 marks]*

      ii. Most recently used *[2 marks]*

      iii. Belady's optimal algorithm *[2 marks]*

3. (a) In a scheduling system known as preemptive priority scheduling, the priority of each process is a (positive or negative) number that changes dynamically, at a rate α when it is on the ready queue, and at a rate β when it is actually running. Processes are initialised with priority 0, and reset to priority 0 when they return from a blocked state. (Positive α means increasing priority.) What happens if the algorithm is run with the following settings? Explain your answers.

   i. $\beta > \alpha > 0$ *[3 marks]*

   ii. $\beta < \alpha < 0$ *[3 marks]*

   (b) What is the role of the boot block, superblock and inode area in the original Unix system *[4 marks]*

   (c) Consider a file system with 512B blocks of file data where each inode contains 13 block pointers each of which takes up 4B. 10 of the inode pointers are direct pointers with the remaining 3 pointers as follows: 1 single indirect pointer, 1 double indirect pointer and 1 triple indirect pointer.

   i. What is the size of the inode? *[1 mark]*

   ii. What is the size of the largest file that can be stored? *[3 marks]*

   (d) Assume a single-platter disk with the following characteristics. It has 200 tracks (number 1 at the inside, 200 at the outside). Assume the head is stationary at track 100, after moving there from an inner track. The disk request queue contains requests for tracks 55 (front), 58, 39, 90, 160, 38, and 184 (back).

   For each of the algorithms below show the sequence of moves made to service the request queue and calculate the total number of tracks the head moves.

   i. FCFS *[2 marks]*

   ii. SSTF *[3 marks]*

   iii. SCAN *[3 marks]*

   iv. C-LOOK *[3 marks]*