भारतीय प्रौद्योगिकी संस्थान खड़गपुर
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
Department of Computer Science and Engineering

CS 60038: Advances in Operating System Design **End Semester Examination**

Full Marks: 100 Time: 3 hours Autumn, 2023

Note:

(i) There are SEVEN questions in this paper, broken up into PART A and PART B. Answer all the questions. Answer all the parts of a single question together. **Use separate examination copies for PART A and PART B. Write down PART A or PART B on top of the copies.**

(ii) The answers should be precise and to-the-point. Marks will be deducted for unnecessary texts.

(ii) Write down the assumptions clearly, if any. No clarifications will be given during the exam hours.

PART A (FOUR Questions)

1. (a) During a system call execution, how does the trap instruction find out the location of the instructions code-base that needs to be executed corresponding to that system call? Why can't a normal user-space program execute that code-base? [2+2]

   (b) How does a Radix tree from the IDR API help in PID allocation? Explain How this tree structure provides better performance in comparison with a linked list of allocated PIDs. [3]

   (c) Why do we typically implement the Linux device drivers as Loadable Kernel Modules? [2]

   (d) Consider that a Linux process executes a `connect` system call over a socket. What would be the process state (in terms of Linux process states) for this process immediately after executing this call? [1]

2. (a) What is the difference between task and thread in MACH? [1+1]

   (b) Write down two differences between the MACH process priority band and the Linux process priority band. What advantages do the MACH process scheduler get by those two changes that you have mentioned? [1+1]

   (c) The clutch scheduler uses Earliest Deadline First (EDF) Scheduling at the bucket level, but uses an interactivity score-based algorithm at the thread group level. What is the purpose of using these two different algorithms at the two levels? [2]

   (d) What is worst case execution latency (WCEL) in the clutch scheduler? What is its purpose? [2]

3. (a) Consider the guest OS and the VMM for Type 1 virtualization. Where does these two components operate on the protection ring for (i) full virtualization, (ii) Para virtualization, (iii) Hardware-assisted virtualization? Just give your answer, no need to explain. [3]

(b) What is VMCS? With the diagram of the VMM life cycle, explain which parts of the VMCS (Guest state area or Host state area) need to be loaded or saved during each transition of the VMM life cycle (You DO NOT need to explain the VMM life cycle. Just draw the diagram and answer the question with respect to each of the transitions). [1+3]

(c) Explain what is the difference between a shadow page table and Second-level address translation (SLAT) methods to handle two-level paging while resolving a virtual address generated from the guest OS running over a Type 2 hypervisor. [3]

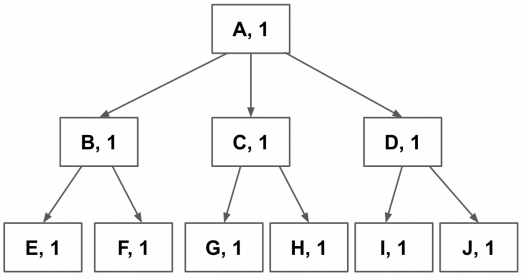
4. (a) Consider that you want to store a file of size 2.4 GB. Assume that the disk block size is 4 KB. How many total inode and block pointers (counting the pointers at each level – direct, single indirect, double indirect, etc.) are needed to point the data blocks for this file from the inode. With a diagram, show the direct, single indirect and double indirect pointers that will be used to point the data blocks. [2+3]

(b) Consider that you want to create a new file named `CS60038.txt` and write 6 KB data to it with a single `write` syscall. Draw a timing diagram to show the read/write operations on the data bitmap, inode bitmap, inode and data blocks for the above operations on a ext2 file system. Assume that the data block is of size 4 KB. You do not need to explain anything; just draw the timing diagram clearly to show all the read/write operations in a correct temporal order. [5]

(c) Consider that you have modified the ext2 file system in such a way that it ensures the data blocks are written/updated only after the inode is updated. Now you open a file in this modified ext2 file system and change a single character in the file such that it only updates the content of an existing data block (no new data block is written). Can this file system be inconsistent if the system crashes while you were performing the update operation? Explain your answer. [2]

(d) Consider $b = 1$ for a COW-friendly B+ Tree. Insert the following keys to that B+ tree in the given order: 2, 5, 8, 9, 10, 12, 17, 20. Ensure that COW-friendly properties are preserved. Show the resultant tree after each step of insertion. You do not need to explain anything, just draw the updated tree at each step. [4]

(e) Consider the COW-friendly B tree as shown in the following figure.



Each node shows the key and the reference count for that page. Now apply the following operations in the given order on this B tree and draw the resultant B tree after performing each of the operations. (i) Clone the tree. Let the root of the clone be M. (ii) Update the pages H and J on the cloned tree with root M. Let the updated pages be H$'$ and J$'$. (iii) Insert a page with key K at the parent of J$'$. (iv) Delete the parent of H$'$. You do not need to explain anything. Just draw the resultant tree with the updated reference counts after each of the above four operations. [1+1+2+2]

PART B (THREE Questions)

5. (a) State two reasons with justification why you think the O(1) scheduler was replaced by CFS in Linux. Be specific, do not describe the algorithms. [3]

   (b) Describe the scheduling algorithm followed for SCHED_RT class in Linux. You are not required to refer to exact code. [4]

   (c) In CFS, the next task to be run is selected as the task with the minimum vruntime. The data structure most suited for selecting the minimum from a set is a priority queue. However, Linux uses a Red-Black tree instead of say, a min-heap. Can you think of some potential reasons that justifies the use of a RB-tree instead of heap in CFS implementation? Justify your answer, but be specific. [3]

6. (a) Define the following with respect to a task to be scheduled in an embedded/real-time operating system: (i) tardiness, (ii) laxity. What is meant by Makespan and how is it calculated? [1 + 1 + 2]

   (b) Consider the periodic tasks A, B, C, and D shown below. The deadline shown is relative deadline.

	A	B	C	D
Execution Time	4	1	2	1
Period	11	9	6	12
Deadline	11	9	6	12

   (i) Does there exist a feasible schedule for the set of tasks if rate monotonic scheduling is followed? Justify your answer.

   (ii) Does there exist a feasible schedule for the set of tasks if EDF scheduling is followed? Justify your answer.

   (iii) Show the schedules (as a Gantt chart, upto time 30) obtained on applying rate monotonic scheduling and EDF scheduling on the set of tasks. Just show the chart with times and tasks marked clearly, no explanation is needed. Assume that all tasks arrive for the first time at time 0 itself.

   (iv) In general, name one advantage and one disadvantage of using EDF scheduling over rate monotonic scheduling.

   [3 + 2 + 6 + 3]

   (c) What is the basic difference between using VxWorks and Wind River Linux even though both are products from the same company (Wind River Software)? [2]

7. (a) The incomplete version of Maekawa's algorithm taught in class clearly has deadlock. How can you handle it to make it deadlock-free? Assume that I know the algorithm up to the point taught in class, start from after that. Specify the messages sent and actions taken on their receipt clearly. [5]

   (b) Consider the distributed mutual exclusion algorithm taught in class. Clearly show an example with not more than 5 nodes as to how an outdated request for critical section from one node can arrive at another node. List the sequence of events that can cause it precisely, do not write long descriptions. [3]

   (c) In the context of DFS, state one advantage of a stateless protocol over a stateful protocol (explain with a clear example). Is NFSv3 a stateful or stateless protocol? Justify in 1-2 sentences. What is meant by sharing semantics in the context of a DFS? [3 + 2 + 2]

(d) Consider a system with a large number nodes distributed over a wide area network. All the nodes are homogeneous, meaning they have the same CPU, memory etc. Users can give programs at any of the node to run at any time. However, as a system designer, you want the total load of the system to be more or less distributed over all the nodes as much as possible. Can you design a protocol that will try to achieve this? Assume that the load a of a single node refers to its CPU utilization only. At the end, briefly comment on the pros and cons of your designed protocol. [5]