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

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

Full Marks: 100 Time: 2 hours Autumn, 2023

Note:

(i) There are THREE questions in this paper. Answer all the questions. Answer all the parts of a single question together.

(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.

1. (a) Explain how the microkernel architecture differs from the monolithic kernel architecture (with a figure). Answer briefly and precisely. [4]

   (b) Do you need a context switch when a process changes its mode from user mode to kernel mode (consider Linux processes)? Explain your answer in brief. [4]

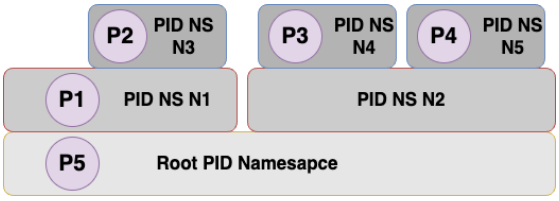
   (c) To initiate a system call, the user-space program typically calls a stub function rather than a actual system call function – what is the purpose for the same? [4]

   (d) With an example, write down the execution steps that happen when a user-space program calls a system call stub function. Answer briefly and precisely. Mark which steps in the execution are performed on the software and which steps are performed on the hardware. [8]

2. (a) What is the difference between `TASK_UNINTERRUPTIBLE` and `TASK_INTERRUPTIBLE` process states? Assume that a process is in `TASK_UNINTERRUPTIBLE` state. What is mode for this process – user or kernel? Explain your answer. [3+2]

   (b) The `task_struct` structure contains two fields corresponding to the process IDs – `pid_t pid` and `struct pid *thread_pid`. What is the purpose of maintaining these two notions of process IDs? [5]

   (c) Consider a PID namespace (NS) hierarchy as shown in the figure below.



Two PID namespaces N1 and N2 are created on top of the root PID namespace. PID namespace N3 runs on top of N1, and namespace N4 and N5 run on top of N2. There are five processes P1 to P5, running over the respective namespaces as shown in the figure.
How many process IDs will be allocated for each of the above five processes? Considering that the process Pi gets the IDs as Pi_1, Pi_2, ..., which of the process IDs can have identical neumeric values? [5+5]

(d) When you make a `fork` system call, the memory for the `struct pid` structure for the newly created child process is allocated from the kernel memory cache. What is the reason for using the cache? [3]

(e) Explain the steps of how a neumeric process ID is allocated to a new child process. Answer briefly and precisely. [5]

(f) What is the difference between `fork()` and `vfork()` to create a new child process? What is a kernel thread? [4+3]

3. (a) Describe the mechanisms used to favor interactive jobs over batch jobs in case of (i) the O(1) scheduler, and (ii) Completely Fair Scheduler (CFS). Your answer should be to the point and brief. [5]

(b) Consider a single-CPU Linux system with 6 fair-class tasks with priority 110, 115, 118, 122, 128, and 134. All the tasks are in `TASK_RUNNING` state. The `vruntime` values stored for the tasks at some point of time are 17, 25, 30, 18, 22, and 35 milliseconds respectively. The task with priority 110 is running currently on the CPU. The scheduler tick occurs every 1 millisecond. Which task will be running in the CPU after the next scheduler tick? What will be the timeslice assigned to it? What will be the value of the `min_vruntime` after processing the next scheduler tick if the value after processing the current tick was 20? Show all calculations and justify your answers briefly for each part. Use the weight matrix given below.

```
static const int prio_to_weight[40] = {
/* -20 */     88761,     71755,     56483,     46273,     36291,
/* -15 */     29154,     23254,     18705,     14949,     11916,
/* -10 */      9548,      7620,      6100,      4904,      3906,
/*  -5 */      3121,      2501,      1991,      1586,      1277,
/*   0 */      1024,       820,       655,       526,       423,
/*   5 */       335,       272,       215,       172,       137,
/*  10 */       110,        87,        70,        56,        45,
/*  15 */        36,        29,        23,        18,        15,
};
```

[3+5+2]

(c) What is the use of keeping the `min_vruntime` in the `cfs` queue? Why do you think the fact that it increases monotonically is not a problem given its purpose? Your answer should be to the point and brief. [2+3]

(d) Why are there three fields kept in the `task_struct` of a process for storing priority of a process? Answer briefly. [5]

(e) Why are the different `sched_class` variables stored as an array in Linux? Justify your answer briefly and precisely. [5]

(f) On a scheduler tick, how does the kernel know whether the currently running process needs to be rescheduled or not? Your answer should be specific to the Linux code (if you do not remember exact variable/function names, that's ok, but you need to specify where are the relevant variables kept, what do they contain, how they are checked etc.). [7]

(g) Suppose a device uses a wait queue that currently has 3 entries with `WQ_FLAG_EXCLUSIVE` flag set to false and 5 entries with `WQ_FLAG_EXCLUSIVE` flag set to true. The first three entries are in `TASK_UNINTERRUTIBLE` state and the next 5 are in `TASK_INTERRUPTIBLE` state. Let us name the entries w1, w2, w3, ..., w7, w8 starting from the head to the tail of the queue. The constant `WAITQUEUE_WALK_BREAK_CNT` that limits the max number to be woken up in one call of `__wake_up_common()` function is set to 4. After one call to `___wake_up_common()` function that traverses the wait queue to wake up processes, (i) what entries will be there in the wait queue? (ii) for each entry, what will be the state of the corresponding process and what will be the flag field value in the `wait_queue_entry`? Justify your answer at each step briefly. If you make any assumptions, state them clearly. [4+4]