Yash Singhal 2400290120285 CS-D

1. CPU Scheduling Policies Implemented

Each scheduling policy in Linux corresponds to a file in kernel/sched/:
- fair.c → Completely Fair Scheduler (CFS) for normal tasks.
- rt.c → Real-Time Scheduler (SCHED_FIFO and SCHED_RR).
- deadline.c → Earliest Deadline First Scheduler (SCHED_DEADLINE).
- idle.c → Idle task scheduler.
- ext.c, ext_idle.c → Scheduler extensions.
- core.c, core_sched.c → Core scheduling logic.
- autogroup.c → Per-user fair scheduling.
- isolation.c → CPU isolation for dedicated workloads.

2. Comparison with Classical Scheduling Algorithms

- FCFS: Similar to base fairness in CFS.
- SJF/SRTF: CFS favors tasks with less runtime (smaller vruntime).
- Priority Scheduling: Used in RT and Deadline schedulers.
- Round Robin: Implemented in rt.c (SCHED_RR) for real-time tasks.
- Multilevel Queue: Separate scheduling classes act as queues with priority hierarchy.

3. Round Robin (SCHED_RR) Implementation in rt.c

- Located in linux/kernel/sched/rt.c.
- Each task has a time slice (rr_timeslice).
- Function task_tick_rt() decrements time slice on each tick.
- When expired, task is requeued and context switch is triggered.

Simplified logic:

```
if (--p->time_slice == 0) {
   p->time_slice = RR_TIMESLICE;
   requeue_task_rt(rq, p, 0);
   set_tsk_need_resched(p);
}
```

4. Data Structures Used in Scheduling

- task_struct: Represents each process.
- rq: Runqueue per CPU for runnable tasks.

- cfs_rq: Runqueue for CFS tasks (red-black tree).
- rt_rq: Runqueue for RT tasks.
- dl_rq: Runqueue for Deadline tasks.
- rb_tree: Stores CFS tasks by virtual runtime.
- plist: Priority list used by RT and Deadline schedulers.

5. Disk Schedulers Supported

Located in linux/block/:
- noop-iosched.c → FIFO-based.
- deadline-iosched.c → Ensures fairness and low latency.
- cfq-iosched.c → Fair queueing (deprecated).
- bfq-iosched.c → Budget Fair Queueing.
- mq-deadline, kyber → Multi-queue schedulers for NVMe/SCSI.

6. Comparison with Basic Disk Scheduling Algorithms

- FCFS → noop scheduler.
- SSTF → deadline scheduler.
- SCAN / C-SCAN → cfq or bfq schedulers.
- LOOK / C-LOOK → mq-deadline scheduler.

7. Paging, Page Replacement, and Memory Allocation

Handled in mm/ directory:
- mm/memory.c, mm/mmap.c → Paging and virtual memory.
- mm/vmscan.c → Page replacement (LRU).
- mm/page_alloc.c → Buddy allocator.
- mm/slab.c, mm/slub.c → Kernel object allocation.

8. Purpose of vmscan.c

vmscan.c handles page reclaiming to free memory under pressure.
Functions:
- Scans active/inactive LRU lists.
- Reclaims or swaps out pages.
- Works with kswapd to maintain free pages.
Purpose: Manage memory efficiently and prevent out-of-memory conditions.