Homework 5

CS211 - Fall 2024

(You should try to answer first and then compare your solution with the ChatGPT solution)

This homework is designed to help you become familiar with scheduling concepts and algorithms in operating systems

1. Let P_1, P_2, P_3, P_4 be processes with execution times of 53, 8, 68, and 24, respectively. Find the average waiting time and average completion time for the following scheduling scenarios: the best-case FCFS, the worst-case FCFS, and Round-Robin with quantum values of 1, 5, 8, 10, and 20.

2. Why does Shortest Remaining Time First (SRTF) scheduling help achieve the best response time and I/O throughput? Provide an example to illustrate your explanation.

3. What is the CPU scheduling policy of Linux? Briefly describe this policy and explain its strengths and weaknesses (e.g., in terms of response time, fairness, overhead, etc.)

① Best case FCFS: P_2, P_4, P_1, P_3

- avg waiting time: $\frac{0+8+(8+24)+(8+24+53)}{4} = 31.25$
- avg completion time: $\frac{8+(8+24)+(8+24+53)+(8+24+53+68)}{4} = 69.5$

Worst case FCFS: P_3, P_1, P_4, P_2

- avg waiting time: $\frac{0+68+(68+53)+(68+53+24)}{4} = 83.5$
- avg completion time: $\frac{68+(68+53)+(68+53+24)+(68+53+24+8)}{4} = 121.75$

Round-robin assuming FCFS order is: P_1, P_2, P_3, P_4

RR quantum = 1:

- avg waiting time: $\frac{0+1+2+3}{4} = 1.5$
- avg completion time: $\big((53+8+52+24)+(8+8+7+7)+(68+53+8+24)+(24+8+24+24)\big)/4 = 100$

RR quantum = 5:

- avg waiting time: $\frac{0+5+(5+5)+(5+5+5)}{4} = 7.5$
- avg completion time: $\big((53+8+50+24)+(8+10+5+5)+(68+53+8+24)+(24+25+8+25\big)/4 = 99.5$

RR quantum = 8

- avg waiting time: $\frac{0+8+16+24}{4} = 12$
- avg completion time: $\big((53+8+48+24) + (8+8) + (68+53+8+24) + (24+24+8+24)\big) \,/\, 4 = 95.5$

RR quantum = 10:

- avg waiting time: $\frac{0+10+(10+8)+(10+8+10)}{4} = 14$
- avg completion time: $\big((53+8+50+24) + (10+8) + (68+53+8+24) + (24+30+8+30)\big) \,/\, 4 \approx 99.5$

RR quantum = 20:

- avg waiting time: $\frac{0+(20+8)+(20+8+20)}{4} = 19$
- avg completion time: $\big((53+8+40+24) + (20+8) + (68+53+8+24) + (24+8+40+40)\big) \,/\, 4 = 104.5$

② Avg response time: $\frac{\sum_{i}^{n} (\text{Time job i gets started} - \text{Time job i enters queue})}{n}$

Avg I/O throughput: $\frac{\text{Number of jobs done}}{\text{Unit of time}}$ or $\frac{\text{Number of all jobs (n)}}{\text{Total time to complete all jobs}}$

SRTF prioritize shorter jobs → Reduce overall waiting time across tasks on queue

- Reduce response time theo cấp số cộng (only big jobs suffer delay, but many short jobs can reduce waiting for big jobs to get done)
  ↳ Benefits from this if there are fewer big jobs
- Increase the count of jobs done ⇒ Better throughput

Example: P_1, P_2, P_3, P_4 with duration 53, 8, 68, 24

Assume all jobs enter queue at time $t=0$

	Avg response time	Avg throughput
Vanilla FCFS	$\frac{0+53+8+68}{4} = 32.25$	$\frac{4}{(0+53)+(53+8)+(53+8+68)+(53+8+68+24)}$ $= \frac{4}{396} \approx 1.0101\,\%$
SRTF	$\frac{0+8+24+53}{4} = 21.25$ ↓ Better than FCFS, not as good as RR	$\frac{4}{(0+8)+(8+24)+(8+24+53)+(8+24+53+68)}$ $= \frac{4}{278} \approx 1.4388\,\%$ → Better than FCFS and RR
Round robin quantum = 10	$\frac{0+10+18+28}{4} = 14$	$\frac{4}{99.5 \times 4}$ $= \frac{4}{398} \approx 1.005\,\%$

⇒ SRTF is good but not outstanding :)

(3) Classical alg. with no direct equivalent in linux: SRTF

Classical scheduler	FCFS	Multilevel Queue	RR	Batch scheduling
Linux scheduler	SCHED_FIFO	SCHED_CFS	SCHED_RR	SCHED_BATCH
Response time	Depends (long if big jobs arrive first, short otherwise)	Short (newer jobs has exec. time = 0 → higher priority → get started soon)	Short (wait for at most quantum duration * queue length)	Long (not focus on optimize response time)
Fairness	Doesn't care	Very fair (ensure fair share of CPU, prioritize jobs with fewer exec. time)	Very fair (ensure fair shair of CPU time)	Doesn't care
Overhead	Very little (only a basic queue, no extra context switching)	A lot (Needs a RBTree & priority queue & frequent context switching)	Quite a lot (needs a queue and frequent context switching)	Very little (pre-compute execution order – one off cost)
Throughput	Relatively high (jobs are run to completion, no splitting)	Very low (job gets split, takes forever to done)	Very low (job gets split, takes forever to done)	High (designed to optimize throughput)
Avg-completion time	Short (jobs are run to completion, no split)	Long (jobs get split → takes forever)	Long (jobs get split → takes forever)	Short (jobs are run to completion, no split)
Avg-waiting time (exec time – queue time)	Long (must wait for previous job to done to start)	Short (jobs get started soon after enter queue)	Short (jobs get started soon after enter queue)	Long (must wait for previous job to done to start)

Linux provides ~~4~~ 6 scheduling algorithms:

- SCHED_FIFO: First come first serve, jobs arrive earlier gets started first, jobs are run to completion

- SCHED_FAIR: Completely fair scheduler, Maintains a priority queue (RBTree), jobs with lesser CPU time gets bump to top of queue → Always select job that's been run the least

- SCHED_RR: Round robin algorithm, CPU time is divided into non-overlapping time slice (quantum), then jobs are rotated to run until done (yield) or timeout (preempt by scheduler)

- SCHED_BATCH: Batch scheduling is often used when we know the list of jobs in advance & can organize them into batches. Once sorted out, each batch is run with little to no scheduling modification.
  Jobs are often not split but run to completion.
  ↳ Suitable for jobs like training models on server.

Above information are extracted from this man page:
man7.org/linux/man-pages/man7/sched.7.html

SCHED_OTHER and SCHED_IDLE are not mentioned because I don't quite understand them (& their use case) at the point of writing this

To find quantum duration of RR scheduling

run this

```
/home/lanphgphm❱❱❱ ls /proc/sys/kernel
acct                                    kexec_load_limit_reboot                 perf_event_max_sample_rate              sem_next_id
acpi_video_flags                        keys                                    perf_event_max_stack                    shm_next_id
arch                                    kptr_restrict                           perf_event_mlock_kb                     shm_rmid_forced
auto_msgmni                             max_lock_depth                          perf_event_paranoid                     shmall
bootloader_type                         max_rcu_stall_to_panic                  pid_max                                 shmmax
bootloader_version                      modprobe                                poweroff_cmd                            shmmni
bpf_stats_enabled                       modules_disabled                        print-fatal-signals                     soft_watchdog
cad_pid                                 msg_next_id                             printk                                  softlockup_all_cpu_backtrace
cap_last_cap                            msgmax                                  printk_delay                            softlockup_panic
core_pattern                            msgmnb                                  printk_devkmsg                          split_lock_mitigate
core_pipe_limit                         msgmni                                  printk_ratelimit                        stack_tracer_enabled
core_uses_pid                           ngroups_max                             printk_ratelimit_burst                  sysctl_writes_strict
ctrl-alt-del                            nmi_watchdog                            pty                                     sysrq
dmesg_restrict                          ns_last_pid                             random                                  tainted
domainname                              numa_balancing                          randomize_va_space                      task_delayacct
ftrace_dump_on_oops                     numa_balancing_promote_rate_limit_MBps  real-root-dev                           threads-max
ftrace_enabled                          oops_all_cpu_backtrace                  sched_autogroup_enabled                 timer_migration
hardlockup_all_cpu_backtrace            oops_limit                              sched_cfs_bandwidth_slice_us            traceoff_on_warning
hardlockup_panic                        osrelease                               sched_child_runs_first                  tracepoint_printk
hostname                                ostype                                  sched_deadline_period_max_us            unknown_nmi_panic
hung_task_all_cpu_backtrace             overflowgid                             sched_deadline_period_min_us            unprivileged_bpf_disabled
hung_task_check_count                   overflowuid                             sched_energy_aware                      unprivileged_userns_clone
hung_task_check_interval_secs           panic                                   sched_rr_timeslice_ms                   user_events_max
hung_task_panic                         panic_on_io_nmi                         sched_rt_period_us                      usermodehelper
hung_task_timeout_secs                  panic_on_oops                           sched_rt_runtime_us                     version
hung_task_warnings                      panic_on_rcu_stall                      sched_schedstats                        warn_limit
io_delay_type                           panic_on_unrecovered_nmi                sched_util_clamp_max                    watchdog
io_uring_disabled                       panic_on_warn                           sched_util_clamp_min                    watchdog_cpumask
io_uring_group                          panic_print                             sched_util_clamp_min_rt_default         watchdog_thresh
kexec_load_disabled                     perf_cpu_time_max_percent               seccomp                                 yama
kexec_load_limit_panic                  perf_event_max_contexts_per_stack       sem
```

RR quantum

scheduling specs are here

```
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_autogroup_enabled
1
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_cfs_bandwidth_slice_us
5000
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_child_runs_first
0
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_deadline_period_max_us
4194304
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_deadline_period_min_us
100
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_rr_timeslice_ms
100
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_rt_period_us
1000000
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_rt_runtime_us
950000
/home/lanphgphm❱❱❱ cat /proc/sys/kernel/sched_schedstats
0
/home/lanphgphm❱❱❱
```

quantum of RR is 100 ms