SYSC 4001 Operating Systems Fall 2025

Assignment 3 - Part 1 Report

SYSC4001 L3 – Group 3

Gabriel Wainer

Muhammad Ali – 101291890

Gregory Horvat – 101303925

https://github.com/aliooo36/SYSC4001_A3_P1.git

EP Scheduler:

Basic Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.009901	100.0	0.0	0.0
Test 2	0.019802	75.0	25.0	25.0
Test 3	0.009950	125.0	25.0	25.0
Test 4	0.016575	113.33	53.33	53.33
Test 5	0.033058	60.0	30.0	30.0
Test 6	0.033113	70.0	40.0	40.0

Across the six EP basic tests, the scheduler consistently executed processes strictly by priority, producing predictable non-preemptive behaviour. Tests with a single high-priority job (Test 1) showed minimal wait and response times, while multi-process tests (Tests 2-4) demonstrated how lower-priority processes accumulate delay until all higher-priority tasks finish. This is most visible in Test 3, where the late-arriving second process increases overall turnaround and lowers throughput, showing a common drawback of non-preemptive priority scheduling. In contrast, Tests 5 and 6, where all processes share the same priority and therefore follow FCFS, achieved the highest throughputs due to shorter total execution windows and staggered arrivals, though wait and response times gradually rose as more processes joined the queue. Overall, the EP scheduler performs well when priorities align with arrival patterns but shows increased turnaround and reduced responsiveness when high-priority processes arrive after lower-priority ones.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	10	8	7
Test 3	8	2	7
Test 4	18	2	16
Test 5	33	15	30
Test 6	33	2	30

Across all six EP basic tests, memory usage remained consistent with the fixed-partition layout, and fragmentation levels were directly tied to which partitions incoming processes occupied. Smaller processes frequently ended up in significantly larger partitions, creating substantial internal fragmentation, most notably in Tests 5 and 6, where multiple 1 MB processes were placed in 15 MB, 10 MB, and 8 MB partitions, resulting in peak fragmentation levels of 30 MB. Tests with fewer processes (such as Test 1 and Test 3) naturally produced lower memory usage and minimal fragmentation. Meanwhile, Test 4 and especially Tests 5-6 exhibited higher peak memory usage because more processes were loaded concurrently, activating multiple partitions at once.

BrightSpace Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.090909	10.0	0.0	0.0
Test 2	0.083333	11.0	0.0	0.0
Test 3	0.125000	11.0	3.5	3.5
Test 4	0.133333	9.5	1.5	0.0

Across the Brightspace EP tests, execution behaviour remained consistent with non-preemptive priority scheduling, producing minimal wait and response times for single-process workloads and predictable scheduling patterns once additional processes were introduced. Test 1 and Test 2 both completed with zero waiting since no competing process arrived before completion, with Test 2 showing a slightly longer turnaround due to the inserted I/O operation. In Tests 3 and 4, the presence of a second process caused turnaround, wait, and response times to increase, driven entirely by the arrival time of the short secondary job relative to the primary process's CPU or I/O phases. Test 4 achieved the highest throughput because the short job completed while the main process was blocked on I/O, showing how EP can use idle CPU cycles when I/O-bound behaviour is present. Overall, the execution metrics highlight that EP handles basic CPU/I/O interactions efficiently but offers no mechanism to reduce waiting caused by later-arriving lower-priority tasks.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	2	2	1
Test 3	8	2	6
Test 4	10	2	7

Memory behaviour in the Brightspace tests remained highly stable for the single-process cases, where only the smallest partition was used and internal fragmentation was therefore minimal. Fragmentation and total memory usage increased only when a second process entered the system, as seen in Tests 3 and 4, where the allocator was forced to place processes into larger partitions due to fixed-partition constraints. This resulted in noticeably higher internal fragmentation, up to 6 MB in Test 3 and 7 MB in Test 4, even though both processes themselves were small. Test 4 also temporarily reached the highest memory usage because both processes coexisted between I/O transitions, activating two partitions concurrently.

ExceededMemorySize Tests:

Execution Results:

Test	Processes Loaded	Processes Rejected	Throughput (proc/ms)	Avg Turnaround	Avg Wait	Avg Response
Test 1	0	1	1.000000	0.0	0.0	0.0
Test 2	0	2	2.000000	0.0	0.0	0.0
Test 3	0	3	3.000000	0.0	0.0	0.0
Test 4	0	4	1.000000	0.0	0.0	0.0
Test 5	0	5	4.000000	0.0	0.0	0.0

The ExceededMemorySize tests all demonstrate the scheduler's correct handling of processes whose memory requirements exceed the fixed partition limits. In every case, processes were immediately terminated at time 0 without ever entering the READY state, resulting in zero turnaround, wait, and response times across all tests. Throughput values appear artificially high because the simulator counts each rejected process as completed at time 0; however, no actual scheduling or CPU execution took place. This behaviour confirms that the memory check occurs before admission to the ready queue and that EP handles oversized processes deterministically by rejecting them instantly.

Memory Results:

Because every process in these tests exceeded the maximum available partition size, no memory allocation was ever performed, resulting in all associated memory report files remaining empty. This outcome correctly reflects the system's admission control logic: memory validation occurs before allocation, meaning oversized processes are rejected without attempting to assign them to a partition or modify the memory layout. As a result, memory usage remained effectively zero, fragmentation did not occur, and no partitions were activated.

IOBound Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.027027	36.0	0.0	0.0
Test 2	0.021978	85.0	32.5	7.5
Test 3	0.018868	112.67	54.33	31.67
Test 4	0.017316	181.25	116.0	80.0
Test 5	0.014749	212.60	139.20	99.0

Across the five I/O-bound tests, execution behaviour under EP showed increasingly complex interactions between CPU bursts, I/O wait periods, and strict priority ordering, leading to larger turnaround, wait, and response times as more processes were introduced. In the single-process case (Test 1), the job alternated cleanly between CPU and I/O phases with no waiting, while Test 2 showed how competition for the CPU during overlapping I/O cycles increased both processes' response and wait times. Tests 3 through 5 exhibited queueing delay under EP: higher-priority processes repeatedly preempted lower-priority ones upon returning from I/O, causing response and wait times to grow substantially, reaching over 100 ms of average wait by Test 5. Throughput consistently decreased across the test suite as total execution timelines lengthened due to frequent I/O stalls and the growing ready queue.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	10	8	7
Test 3	20	2	16
Test 4	35	15	27
Test 5	60	8	51

Memory behaviour in the I/O-bound tests reflected the cumulative effect of multiple small processes occupying partitions whose sizes were much larger than the processes themselves, leading to rising internal fragmentation as additional processes were introduced. Test 1 produced minimal fragmentation because only the smallest partition was used, but fragmentation increased sharply beginning in Test 2 as two processes were forced into the 8 MB and 2 MB partitions, generating up to 7 MB of waste. Tests 3 through 5 amplified this pattern: more processes meant more simultaneously occupied partitions, causing internal fragmentation to reach 16 MB in Test 3, 27 MB in Test 4, and a peak of 51 MB in Test 5 when nearly all partitions were filled.

RR Scheduler:

Basic Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.009901	100.0	0.0	0.0
Test 2	0.019802	75.0	25.0	25.0
Test 3	0.006645	250.0	100.0	25.0
Test 4	0.008310	320.0	200.0	80.0
Test 5	0.012461	185.0	105.0	105.0

The RR basic tests highlight how time-slicing distributes CPU access more evenly but increases total turnaround and wait times as the number of processes grows. Test 1 behaves identically to EP since there is only one process, while Test 2 begins to show the interleaving of RR as each process receives CPU in alternating quanta, resulting in higher wait and response times compared to a non-preemptive run. Tests 3 through 5 show us the growing cost of context switching and round-robin queue rotation: later-arriving processes spend substantial time waiting for their turn in the cycle, inflating both average wait and turnaround times, with Test 4 and Test 5 showing especially large delays due to long bursts and multiple processes competing for the CPU. Throughput varies across tests depending on total completion time but generally decreases as the system handles more processes and more context-switch overhead. Overall, RR ensures fairness but exhibits larger delay metrics compared to EP in CPU-bound workloads.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	10	8	7
Test 3	10	2	8
Test 4	20	2	17
Test 5	33	2	30

Memory usage across the RR basic tests remains consistent with fixed-partition allocation, where process size has little influence on overall memory consumption because each admitted process occupies the smallest partition that fits. Internal fragmentation is minimal in Test 1 but increases significantly starting in Tests 2 and 3 when multiple processes load simultaneously, forcing them into 8 MB, 10 MB, or 15 MB partitions despite requiring only 1–2 MB of space. Tests 4 and 5 amplify this effect as more partitions become populated at the same time, driving internal fragmentation as high as 30 MB in Test 5. Since RR encourages processes to remain resident in memory for the entire duration of their execution, the allocator frequently keeps several large partitions active, raising both memory usage and fragmentation.

IOBound Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.027027	36.0	0.0	0.0
Test 2	0.021978	85.0	32.5	7.5
Test 3	0.019868	123.33	65.0	8.33
Test 4	0.017316	212.50	147.25	32.50
Test 5	0.015106	258.00	184.60	38.00

Across the RR I/O-bound tests, execution behaviour reflects the increasing cost of round-robin time-slicing as more processes compete for CPU time, made worse by repeated waiting cycles despite none of the processes ever reaching their scheduled I/O events. Test 1 behaves like a simple single-process RR run with no waiting, while Test 2 introduces alternating CPU access that immediately increases the delay metrics. Tests 3 through 5 clearly show a queue buildup: as additional processes arrive with staggered timings, RR repeatedly rotates through a longer ready queue, causing wait and turnaround times to rise sharply and throughput to decrease. Since the I/O trigger points all occur beyond the CPU bursts, no process ever blocks, meaning the ready queue remains populated and each process must cycle through many quantum rotations before completion. This results in steadily increasing average response and wait times.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	10	10	7
Test 3	20	2	16
Test 4	35	15	27
Test 5	60	8	51

Memory behaviour in the RR I/O-bound tests shows that as more processes are introduced, the fixed-partition model forces several small processes into large partitions, steadily increasing internal fragmentation across the test suite. Test 1 uses only the smallest 2 MB partition, but fragmentation jumps to 7 MB in Test 2 once both the 8 MB and 2 MB partitions are occupied. Tests 3 to 5 escalate this further as more processes arrive: memory use climbs from 20 MB up to 60 MB, and internal fragmentation peaks at 51 MB in Test 5 when multiple large partitions are occupied by 1–3 MB processes. Since none of the processes ever block for I/O, they remain resident in memory for their entire lifetime, keeping many partitions active simultaneously and increasing wasted space.

ExceededMemorySize Tests:

Execution Results:

Test	Processes Loaded	Processes Rejected	Throughput (proc/ms)	Avg Turnaround	Avg Wait	Avg Response
Test 1	0	1	1.000000	0.0	0.0	0.0
Test 2	0	2	2.000000	0.0	0.0	0.0
Test 3	0	3	3.000000	0.0	0.0	0.0
Test 4	0	1	1.000000	0.0	0.0	0.0
Test 5	0	4	4.000000	0.0	0.0	0.0

The RR ExceededMemorySize tests all show that processes requiring more memory than the largest available fixed partition are immediately rejected at time 0, without entering the ready queue or receiving any CPU time. As a result, every test reports zero turnaround, wait, and response times, and throughput appears artificially high because the scheduler counts each rejected process as a completed termination event occurring at the initial timestamp. Overall, the execution results confirm that the RR scheduler relies on memory admission checks before scheduling begins and handles oversized processes with instant termination.

Memory Results:

Since none of the oversized processes in these tests could fit into any fixed memory partition, the allocator never committed a partition, leaving all memory files empty and all partitions unused throughout execution. With memory validation occurring prior to allocation, the simulator correctly prevents illegal memory assignments and avoids altering the memory map when a process exceeds partition limits.

Quantum Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.003984	250.00	0.00	0.00
Test 2	0.009950	150.00	50.00	50.00
Test 3	0.006652	400.00	250.00	100.00
Test 4	0.004994	612.50	412.50	112.50
Test 5	0.003331	1300.00	1000.00	200.00

The RR quantum tests clearly show that as the number of processes increases and CPU bursts become larger, turnaround, wait, and response times grow significantly due to the cyclical nature of round-robin scheduling. Test 1 completes with no waiting since only one job is present, while Test 2 already shows delay as two processes alternate through multiple quanta. Tests 3 through 5 demonstrate increasingly long ready-queue rotation cycles, causing processes to wait hundreds of milliseconds between CPU slices, with Test 5 showing queue congestion and wait times of 1000 ms because all five processes arrive at the same time and must continually cycle through the quantum structure. Throughput decreases across the test suite as longer bursts and more competition extend the total completion time. Overall, these results illustrate how RR guarantees fairness but has substantial overhead when quantum rotations involve many CPU-bound processes with long execution requirements.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	10	8	8
Test 3	20	10	17
Test 4	35	2	30
Test 5	60	25	55

Memory usage in the quantum tests shows the cumulative effect of multiple small processes occupying large fixed partitions for the entire duration of long RR execution cycles. Test 1 uses only the smallest 2 MB partition and therefore has minimal fragmentation, but fragmentation jumps sharply beginning in Test 2 when two partitions (8 MB and 2 MB) remain active for the entire run. Tests 3 through 5 magnify this pattern as more processes remain resident for longer periods: fragmentation reaches 17 MB in Test 3, 30 MB in Test 4, and peaks at 55 MB in Test 5 when all major partitions are filled simultaneously. Since RR keeps processes in memory while they wait for future time slices, partitions stay allocated far longer than in earlier test groups, increasing memory waste.

EP-RR Scheduler:

Basic Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.009901	100.00	0.00	0.00
Test 2	0.019802	75.00	25.00	25.00
Test 3	0.006645	225.00	75.00	0.00
Test 4	0.008310	240.00	120.00	0.00
Test 5	0.012461	200.00	120.00	0.00

Across the EP-RR basic tests, execution behaviour shows us the hybrid nature of the scheduler: jobs are grouped and admitted based on priority, but then share CPU time in a round-robin manner within each priority level. Test 1 behaves like both EP and RR because only one process is present, while Test 2 shows traditional RR alternation because both jobs share the same priority. In Tests 3 through 5, staggered arrivals cause the highest-priority job in each test to run immediately, with later arrivals joining the same queue and rotating through CPU quanta once they become ready. This leads to growing wait and turnaround times as more processes occupy the queue, though the strict EP ordering means new arrivals never preempt a running job, they only participate in RR cycles after the current quantum completes. Overall, the results demonstrate that EP-RR balances fairness and priority grouping well but still inherits RR's increasing delay as queue length grows.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	10	2	7
Test 3	10	2	8
Test 4	20	2	17
Test 5	35	2	31

Memory behaviour in the EP-RR basic tests follows the pattern of fixed-partition allocation, where small processes occupy large partitions and generate internal fragmentation as more processes become resident. Test 1 uses only the smallest partition, but fragmentation increases sharply beginning in Test 2 and Test 3 when multiple processes occupy the partitions despite requiring only 1-2 MB of space. In Tests 4 and 5, additional processes lead to 20-35 MB of active memory usage and fragmentation reaching up to 31 MB as more large partitions remain allocated simultaneously during extended RR cycles. Because EP-RR keeps processes resident throughout their CPU rotations and only removes them once they terminate, memory usage remains high for the full duration of the schedule.

IOBound Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.027027	36.00	0.00	0.00
Test 2	0.021978	87.50	35.00	5.00
Test 3	0.019355	106.67	48.33	0.00
Test 4	0.017167	151.75	86.50	52.50
Test 5	0.015015	184.00	110.60	0.00

The EP-RR I/O-bound tests show that, despite being labeled as I/O-heavy workloads, none of the processes actually reach their programmed I/O points, causing all execution behavior to revert to pure CPU scheduling controlled by the EP-RR hybrid model. Test 1 behaves like a simple single-process priority case, while Test 2 introduces alternating RR behavior between two equal-priority jobs, increasing delay metrics. In Tests 3 through 5, staggered arrivals combined with descending priority levels cause higher-priority processes to run first, with lower-priority jobs waiting until the ready queue at their level becomes active; this layering contributes to rising turnaround and wait times. As more processes enter the system, response and wait times grow because lower-priority processes cannot progress until all higher-priority work in the EP is completed, and within each level, round-robin rotation adds further delay.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	10	2	7
Test 3	20	2	16
Test 4	35	8	27
Test 5	60	8	51

Memory behavior across the EP-RR I/O-bound tests follows the fixed-partition pattern seen in previous tests: as more processes are admitted, increasingly large partitions are used, and internal fragmentation grows substantially even though processes are small. Test 1 occupies only the smallest 2 MB partition, but fragmentation increases immediately in Test 2 where the 8 MB and 2 MB partitions remain active throughout execution. Tests 3 through 5 expand memory usage further, with Test 4 using partitions up to 15 MB and Test 5 reaching full utilization of the 25 MB and 40 MB regions as more processes arrive and remain resident for the entire scheduling cycle. Because none of the processes ever block for I/O, partition occupancy persists for long periods, increasing internal fragmentation and pushing the maximum waste as high as 51 MB in Test 5.

ExceededMemorySize Tests:

Execution Results:

Test	Processes Loaded	Processes Rejected	Throughput (proc/ms)	Avg Turnaround	Avg Wait	Avg Response
Test 1	0	1	1.000000	0.0	0.0	0.0
Test 2	0	2	2.000000	0.0	0.0	0.0
Test 3	0	3	3.000000	0.0	0.0	0.0
Test 4	0	1	1.000000	0.0	0.0	0.0
Test 5	0	4	4.000000	0.0	0.0	0.0

Across all five EP-RR ExceededMemorySize tests, every process requesting more memory than the largest available fixed partition is rejected immediately at time 0, before entering any EP queue or RR rotation cycle. Because memory admission is the first step of the scheduler, processes never reach the READY state and therefore never execute, resulting in zero turnaround, wait, and response times for all tests. Throughput appears unusually high because the simulator counts each rejected process as a completed termination event occurring at the initial timestamp. Since no valid process is ever admitted, neither EP priority ordering nor RR quantum behavior plays any role in these tests.

Memory Results:

Since all oversized processes fail the initial memory-admission check, the memory allocator never assigns a partition in any of the tests, leaving every memory state file empty and all partitions unused throughout execution. No allocation means no fragmentation, no internal waste, and no recorded changes in memory usage. This behavior correctly reflects the fixed-partition model, where any process exceeding the largest partition cannot be placed in memory under any circumstance.

Priority Tests:

Execution Results:

Test	Throughput (proc/ms)	Avg Turnaround (ms)	Avg Wait (ms)	Avg Response (ms)
Test 1	0.003984	250.0	0.00	0.00
Test 2	0.009950	150.0	50.0	0.00
Test 3	0.006652	300.0	150.0	150.0
Test 4	0.004994	500.0	300.0	0.00
Test 5	0.003331	900.0	600.0	240.00

Across the EP-RR priority tests, the hybrid scheduler demonstrates its behavior of grouping processes by priority while applying round-robin scheduling within each priority level. Test 1 shows the single-process case with zero wait and response times, identical to both EP and RR. Test 2 reveals the scheduler's ability to handle same-priority processes with RR fairness, resulting in balanced execution but increased wait times. Tests 3-5 highlight the interaction between priority-based admission and RR rotation, where higher-priority processes (lower PIDs) consistently run first, but within priority levels, processes experience the cyclical delays of RR scheduling.

Memory Results:

Test	Max Memory Used (MB)	Min Memory Used (MB)	Max Internal Fragmentation (MB)
Test 1	2	2	1
Test 2	10	2	8
Test 3	20	2	17
Test 4	35	2	30
Test 5	60	2	55

Memory behavior in the EP-RR priority tests follows the established pattern of fixed-partition allocation, where internal fragmentation grows as more small processes occupy large partitions. Test 1 uses only the smallest 2MB partition, resulting in minimal fragmentation. However, as additional processes enter the system in Tests 2-5, the allocator is forced to place 1-2MB processes into larger partitions, driving internal fragmentation from 8MB in Test 2 to a peak of 55MB in Test 5. The EP-RR scheduler's tendency to keep processes resident throughout their extended execution cycles (due to RR time-slicing) increases this fragmentation, as partitions remain allocated for the duration of each process's lifetime rather than being freed immediately upon completion.