SYSC 4001A – Assignment 3 – Part 1 Report

Atik Mahmud 101318070 ---- Jonathan Gitej 101294584

Test Cases:

Input files 1-4 are simple sequential processes

Input files 5-8 are I/O intensive

Input files 9-12 are CPU intensive

Input files 13-16 are mixed both I/O and CPU)

Input files 17-20 are short processes with I/O

Calculations:

External Priorities (EP)

Output files	Throughput	Average Wait Time	Average Turnaround Time	Average Response Time
execution1.txt	0.06383	14	29.666667	14
execution2.txt	0.046154	15	36.666667	15
execution3.txt	0.055556	15.333333	33.333333	15.333333
execution4.txt	0.058824	13	30	13
execution5.txt	0.004687	176.666667	430	100
execution6.txt	0.005137	205	444.666667	141.666667
execution7.txt	0.004687	213.333333	498.333333	100
execution8.txt	0.004687	234.666667	520	35
execution9.txt	0.005	183.333333	383.333333	183.333333
execution10.txt	0.005	260	460	260
execution11.txt	0.005769	166.666667	340	166.666667
execution12.txt	0.004839	170	376.666667	170
execution13.txt	0.027273	29	68.333333	23.333333
execution14.txt	0.006383	145	345	88.333333
execution15.txt	0.011538	98.333333	206.666667	50
execution16.txt	0.007109	162.666667	327.333333	76.666667
execution17.txt	0.025862	30	80.333333	23.333333
execution18.txt	0.015957	48.333333	124.666667	32.666667
execution19.txt	0.02	40.666667	98	27.666667
execution20.txt	0.01875	37	100.333333	27.666667

Round Robin (RR)

Output files	Throughput	Average Wait Time	Average Turnaround Time	Average Response Time
execution1.txt	0.06383	14.666667	30.333333	14.666667
execution2.txt	0.046154	28.333333	50	28.333333
execution3.txt	0.055556	19.333333	37.333333	19.333333
execution4.txt	0.058824	18	35	18
execution5.txt	0.005172	250	503.333333	36.666667
execution6.txt	0.005556	260.333333	500	51.666667
execution7.txt	0.004545	270	555	41.666667
execution8.txt	0.005	254.666667	540	46.666667
execution9.txt	0.005	350	550	100
execution10.txt	0.005	326.666667	526.666667	100
execution11.txt	0.005769	320	493.333333	100
execution12.txt	0.004839	386.666667	593.333333	100
execution13.txt	0.03	30.666667	70	15
execution14.txt	0.00625	175	375	60
execution15.txt	0.010714	90	198.333333	40
execution16.txt	0.007282	172.666667	337.333333	53.333333
execution17.txt	0.029412	41.666667	92	9.333333
execution18.txt	0.01875	79	155.333333	14.666667
execution19.txt	0.023077	61	118.333333	12.666667
execution20.txt	0.021429	58	121.333333	10.666667

Round Robin + External Priorities (RR + EP)

Output files	Throughput	Average Wait Time	Average Turnaround Time	Average Response Time
execution1.txt	0.06383	14	29.666667	14
execution2.txt	0.046154	15	36.666667	15
execution3.txt	0.055556	15.333333	33.333333	15.333333
execution4.txt	0.058824	13	30	13
execution5.txt	0.004511	148.333333	425	86.666667
execution6.txt	0.00491	164	440.333333	88.333333
execution7.txt	0.004	153.333333	471.666667	48.333333
execution8.txt	0.004438	178.666667	482.666667	65
execution9.txt	0.005	183.333333	383.333333	183.333333
execution10.txt	0.005	260	460	260

execution11.txt	0.005769	166.666667	340	166.666667
execution12.txt	0.004839	170	376.666667	170
execution13.txt	0.027273	28.333333	67.666667	23.333333
execution14.txt	0.005556	116.666667	326.666667	88.333333
execution15.txt	0.010909	103.333333	211.666667	73.333333
execution16.txt	0.007317	120	292.666667	46.666667
execution17.txt	0.024194	27.666667	84.666667	10
execution18.txt	0.014563	39.333333	125	17.333333
execution19.txt	0.019737	30	97	17.666667
execution20.txt	0.018182	30	100.666667	16.666667

Average values:

	External Priorities (EP)	**Round Robin (RR)**	**External Priorities + Round Robin (EP+RR)**
Throughput	0.0198521	0.02060795	0.0195281
Average Wait Time	112.9	160.3333334	98.84999995
Average Turnaround Time	246.6666667	294.0999999	240.7666668
Average Response Time	78.18333335	43.6333334	70.94999995

Verdict:

Simple sequential processes

For the simple input files 1 to 4, all three schedulers give almost the same performance. Throughput is identical for EP, RR, and EP plus RR, and the differences in average wait time, turnaround time, and response time are very small. EP and EP plus RR have slightly lower wait and turnaround times than RR, but the gap is minor. Since there is no need for preemption and no I/O waiting patterns that favor any specific policy, all three schedulers can be considered equally suitable for this group, with a very small edge for EP and EP plus RR.

I/O intensive processes

For the I/O intensive input files 5 to 8, the effect of preemption and the 100 ms timeout becomes clear. RR and EP+RR both preempt the running process every 100 ms, so when an I/O bound process becomes ready it will wait at most one quantum before it can run. This gives RR the best response times and slightly better throughput than EP in this group. However, RR also shows the worst average wait and turnaround times. EP+RR improves on that by using priorities together with the 100 ms timeout. High priority I/O bound processes are not forced to sit behind long low priority jobs, and they still benefit from preemption. As a result, EP+RR achieves lower average wait and turnaround times than both EP and RR, while keeping response times much better than EP. Overall, EP+RR is the best choice for I/O intensive workloads, with RR as a good option when response time is the only priority.

CPU intensive processes

For the CPU intensive input files 9 to 12, processes spend most of their time running on the CPU and rarely block for I/O. In this situation the 100 ms timeout of RR causes many context switches, which increases overhead and stretches both wait time and turnaround time. The data shows that RR has clearly larger wait and turnaround values than the other schedulers in this group, even though its response times are lower because it keeps cycling through all ready jobs. EP and EP+RR give almost identical results here, since there are few interrupts to trigger priority-based preemption and the 100 ms timeout in EP+RR does not change the order very much. Both EP and EP+RR keep wait and turnaround times much lower than RR. For long CPU bound workloads, EP or EP+RR are therefore preferable, with RR offering no real advantage aside from slightly smaller response times.

Mixed CPU and I/O processes

For the mixed input files 13 to 16, both preemption and priority start to matter. RR, with its 100 ms quantum, gives very good response times and slightly higher throughput, since jobs are frequently rotated on and off the CPU. However, the same rotation often leads to longer waits and larger turnaround times, especially when some jobs have long CPU bursts. EP has better control over which jobs run first, because it always chooses the highest priority process, but it cannot interrupt a long running low priority burst once it has started. EP+RR combines both ideas. It still respects external priorities, but it also uses a 100 ms timeout so that long bursts cannot completely block other jobs. In the mixed group this leads to consistently competitive wait and turnaround times, often the lowest among the three schedulers, while keeping response times close to those of RR. For workloads that mix CPU bound and I/O bound jobs, EP+RR offers the best balance between fast start times for important jobs and reasonable completion times for all jobs.

Short processes with I/O

For the short I/O heavy input files 17 to 20, the 100 ms timeout is especially important. RR preempts frequently, so short processes that return from I/O can reach the CPU very quickly. This gives RR the best response times and the highest throughput in this group. The

downside is that RR again shows higher wait and turnaround times overall. EP suffers because it has no preemption once a process starts running. When a short job returns from I/O, it can be forced to wait until the current CPU burst of another job finishes, even if that burst is much longer than 100 ms. EP+RR improves this situation by using both priority and the 100 ms timeout. Short or high priority jobs that come back from I/O do not wait as long, and no single long burst can hold the CPU for more than one quantum. In the measurements, EP+RR gives the lowest average waits and very competitive turnaround times, with response times only slightly worse than RR. For short processes with frequent I/O, RR is the best if you care mainly about response time, while EP+RR is the best overall compromise.

Bonus:

After running the input_file1.txt for EP, RR, EP+RR, memory_status.txt looked like this:

EP

```
================ MEMORY STATUS ================
Time: 0

Partition | Size | Occupied by
-----------------------------------------
1       | 40 MB | free
2       | 25 MB | free
3       | 15 MB | free
4       | 10 MB | free
5       | 8 MB | free
6       | 2 MB | PID 4

Total Used: 2 MB
Total Free: 98 MB
===============================================

================ MEMORY STATUS ================
Time: 0

Partition | Size | Occupied by
-----------------------------------------
1       | 40 MB | free
2       | 25 MB | free
3       | 15 MB | free
4       | 10 MB | free
5       | 8 MB | PID 7
6       | 2 MB | PID 4
```

Total Used: 10 MB
Total Free: 90 MB
==========

========== MEMORY STATUS ==========
Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | PID 2
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 20 MB
Total Free: 80 MB
==========

========== MEMORY STATUS ==========
Time: 15

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 10 MB
Total Free: 90 MB
==========

========== MEMORY STATUS ==========
Time: 27

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free

4 | 10 MB | free
5 | 8 MB | PID 7
6 | 2 MB | free

Total Used: 8 MB
Total Free: 92 MB
==========

========== MEMORY STATUS ==========
Time: 47

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | free
6 | 2 MB | free

Total Used: 0 MB
Total Free: 100 MB
==========

RR
========== MEMORY STATUS ==========
Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | free
6 | 2 MB | PID 4

Total Used: 2 MB
Total Free: 98 MB
==========

========== MEMORY STATUS ==========
Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 10 MB
Total Free: 90 MB
==========

========== MEMORY STATUS ==========
Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | PID 2
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 20 MB
Total Free: 80 MB
==========

========== MEMORY STATUS ==========
Time: 12

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | PID 2
5 | 8 MB | PID 7
6 | 2 MB | free

Total Used: 18 MB
Total Free: 82 MB
==========

================ MEMORY STATUS ================
Time: 32

Partition | Size | Occupied by
----
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | PID 2
5 | 8 MB | free
6 | 2 MB | free

Total Used: 10 MB
Total Free: 90 MB
==

================ MEMORY STATUS ================
Time: 47

Partition | Size | Occupied by
----
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | free
6 | 2 MB | free

Total Used: 0 MB
Total Free: 100 MB
==

EP+RR

================ MEMORY STATUS ================
Time: 0

Partition | Size | Occupied by
----
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free

5 | 8 MB | free
6 | 2 MB | PID 4

Total Used: 2 MB
Total Free: 98 MB
==========

========== MEMORY STATUS ==========
Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 10 MB
Total Free: 90 MB
==========

========== MEMORY STATUS ==========
Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | PID 2
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 20 MB
Total Free: 80 MB
==========

========== MEMORY STATUS ==========
Time: 12

Partition | Size | Occupied by
---

1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | PID 2
5 | 8 MB | PID 7
6 | 2 MB | free

Total Used: 18 MB
Total Free: 82 MB
==========

========== MEMORY STATUS ==========
Time: 32

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | PID 2
5 | 8 MB | free
6 | 2 MB | free

Total Used: 10 MB
Total Free: 90 MB
==========

========== MEMORY STATUS ==========
Time: 47

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | free
6 | 2 MB | free

Total Used: 0 MB
Total Free: 100 MB
==========

========== MEMORY STATUS ==========

Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | free
6 | 2 MB | PID 4

Total Used: 2 MB
Total Free: 98 MB
==========

========== MEMORY STATUS ==========
Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 10 MB
Total Free: 90 MB
==========

========== MEMORY STATUS ==========
Time: 0

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | PID 2
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 20 MB

Total Free: 80 MB
==========

========== MEMORY STATUS ==========
Time: 15

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | PID 7
6 | 2 MB | PID 4

Total Used: 10 MB
Total Free: 90 MB
==========

========== MEMORY STATUS ==========
Time: 27

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free
5 | 8 MB | PID 7
6 | 2 MB | free

Total Used: 8 MB
Total Free: 92 MB
==========

========== MEMORY STATUS ==========
Time: 47

Partition | Size | Occupied by
---
1 | 40 MB | free
2 | 25 MB | free
3 | 15 MB | free
4 | 10 MB | free

5 | 8 MB | free
6 | 2 MB | free

Total Used: 0 MB
Total Free: 100 MB
==========

For the given input file there are three processes with sizes that all fit comfortably into the fixed partitions, so all three schedulers use the same partitions:

- Process 4 is always placed in the smallest partition of size 2
- Process 7 is always placed in the partition of size 8
- Process 2 is always placed in the partition of size 10

This is why the first three snapshots for all schedulers look exactly the same at time 0. At that moment all three processes have arrived and the total used memory is 20 units while 80 units remain free.

The interesting difference appears later, when processes finish.

- With EP the highest priority process runs until completion, so its partition is freed earlier. You can see this when process 2 and process 4 terminate. The total used memory drops in steps from 20 to 10, then to 8, then finally to 0.
- With RR the CPU time is shared in turns, so each process stays in memory for longer, even when it is not running. The snapshots show that the same partitions stay occupied until later times compared to EP. Memory is freed more slowly, but the allocation pattern is still the same.
- With EP plus RR we again see the same partitions being used, but the times at which partitions are freed are closer to the RR case. The CPU scheduling changes the finish times, so memory stays allocated for different lengths of time, even though the allocation policy itself does not change.

The memory allocation algorithm in the simulator always tries the smallest possible partition first. This is a best fit style strategy implemented by scanning the partition list from the smallest to the largest. From these logs we can see that the memory allocation strategy independent of the scheduler. All three schedulers place each process in the same partition. What changes between EP, RR, and EP plus RR is how long each process remains in memory before termination. EP tends to free memory earlier for some processes, while RR and EP plus RR keep processes resident longer. This confirms that CPU scheduling decisions can affect memory usage over time, even when the allocation algorithm is the same.