Phase 2

• Discussion of the memory allocation capabilities using the following memory allocation policies:

1. First Fit
2. Next Fit
3. Best Fit
4. Buddy System Allocation

When a process is picked by the scheduler to be run, it must pass the memory check to get sure whether the it can be allocated or it must wait till there is enough memory for it, the integration of this is in the next_ family functions, these functions now are responsible to get sure that there is some place in memory for a process if the scheduler wants to picks it, it sends that request to allc(int mem) function to ask for some amount of memory, allc in return replies with '-1' if there is no place in memory of that size and replies by the memory starting location if a place could be found, the allc function chooses the specified memory allcation algorithm and run it (FF,NF,BF,BSA).

Your algorithm explanation and results.

1. First Come First Serve (FCFS)

By imagining that process table as a circular array, the pcb_ind is the index of the last entered process in the process table and the pcb_curr is the index to the currently executing process, it is always the case that pcb_curr is behind or the same as pcb_ind, that assures that the processes will execute by the order of arrival.

2. Shortest Job First (SJF)

As this is a batch system, at first we start by picking the least running time process and after finishing we loop again and pick the least running time process and so on.

3. Preemptive Highest Priority First (HPF)

In this algorithm, we need to check for the highest process in terms of priority and execute it, whether there was a process executing before it or not, that is because this algorithm is preemptive so a process might be stopped to run another.

4. Shortest Remaining Time Next (SRTN)

Same as HPF but the condition over preemption is for the remaining time of the process which is kept track of by the process control block of this very process.

5. Round Robin (RR)

We use the quantum entered by the user to determine how much a process will run when it is picked by the scheduler, when no quantum time is entered by the user it will default to ‘1’ clock cycle (1s).
The technique used is that when a process is running, any processes entering the system will reside at the end of the virtual queue data structure and when the quantum of the running process ends it will get back at the back of the queue.

TL;DR, when a process is stopped, it goes to the far back of the queue, back so that any incoming process at the same clock of the running process being stopped will reside before the stopped process in turn in the virtual queue.

***Note: the virtual queue mentioned is the same array described above, but it needs some functions to manipulate it to act like a queue for round robin to work as expected.**

• Your assumptions.

1- The maximum size of processes in the process table is 1024.
2- A process that takes zero time to execute instantaneously.
3- memory per process is at least 1 block.

• Workload distribution.

1- Mostafa Sobhy: Next Fit
2- Mostafa Magdy: First Fit
3- Zyad Hasan: Best Fit.
4- Omer Yacine: Buddy System Allocation

Test Cases

HPF (First Fit)

Makefile M | procs.m M × | scheduler.log | scheduler.perf | memory.log

phase2 > code > exe > procs.m

You, seconds ago | 1 author (You)

```
#id arrival runtime priority memory
1 4 2 2 120
2 7 6 9 188
3 7 0 3 11
4 10  1 2 236
5 12  2 8 125
```

Makefile M | procs.m M | scheduler.log × | scheduler.perf | memory.log

phase2 > code > exe > scheduler.log

```
At time 4 process 1 started arr 4 total 2 remain 2  wait 0
At time 6 process 1 finished arr 4  total 2 remain 0  wait 0  TA 2  WTA 1.00
At time 7 process 3 started arr 7 total 0 remain 0  wait 0
At time 7 process 3 finished arr 7  total 0 remain 0  wait 0  TA 0  WTA 0.00
At time 7 process 2 started arr 7 total 6 remain 6  wait 0
At time 10  process 2 stopped arr 7 total 6 remain 3  wait 0
At time 10  process 4 started arr 10  total 1 remain 1  wait 0
At time 11  process 4 finished arr 10 total 1 remain 0  wait 0  TA 1  WTA 1.00
At time 11  process 2 resumed arr 7 total 6 remain 3  wait 1
At time 12  process 2 stopped arr 7 total 6 remain 2  wait 1
At time 12  process 5 started arr 12  total 2 remain 2  wait 0
At time 14  process 5 finished arr 12 total 2 remain 0  wait 0  TA 2  WTA 1.00
At time 14  process 2 resumed arr 7 total 6 remain 2  wait 3
At time 16  process 2 finished arr 7  total 6 remain 0  wait 3  TA 9  WTA 1.50
```

Makefile M | procs.m M | scheduler.log | scheduler.perf × | memory.log

phase2 > code > exe > scheduler.perf

```
CPU utilization = 81.25%
Avg WTA = 0.90
Avg Waiting = 0.60
```

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log ×

phase2 > code > exe > memory.log

```
1  At time 4 allocated 120 bytes for process 1 from 0  to 119
2  At time 6 freed 120 bytes for process 1 from 0  to 119
3  At time 7 allocated 11  bytes for process 3 from 0  to 10
4  At time 7 freed 11  bytes for process 3 from 0  to 10
5  At time 7 allocated 188 bytes for process 2 from 0  to 187
6  At time 10  allocated 236 bytes for process 4 from 188  to 423
7  At time 11  freed 236 bytes for process 4 from 188  to 423
8  At time 12  allocated 125 bytes for process 5 from 188  to 312
9  At time 14  freed 125 bytes for process 5 from 188  to 312
10 At time 16  freed 188 bytes for process 2 from 0  to 187
11
```

Shortest Remaining Time Next (Next Fit)

Makefile M | procs.m M | scheduler.log × | scheduler.perf | memory.log

phase2 > code > exe > scheduler.log

```
1  At time 4 process 2 started arr 4 total 0 remain 0  wait 0
2  At time 4 process 2 finished arr 4  total 0 remain 0  wait 0  TA 0  WTA 0.00
3  At time 4 process 3 started arr 4 total 0 remain 0  wait 0
4  At time 4 process 3 finished arr 4  total 0 remain 0  wait 0  TA 0  WTA 0.00
5  At time 4 process 1 started arr 4 total 1 remain 1  wait 0
6  At time 5 process 1 finished arr 4  total 1 remain 0  wait 0  TA 1  WTA 1.00
7  At time 7 process 4 started arr 7 total 1 remain 1  wait 0
8  At time 8 process 4 finished arr 7  total 1 remain 0  wait 0  TA 1  WTA 1.00
9  At time 9 process 5 started arr 9 total 4 remain 4  wait 0
10 At time 13  process 5 finished arr 9  total 4 remain 0  wait 0  TA 4  WTA 1.00
11
```

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log

phase2 > code > exe > procs.m

You, seconds ago | 1 author (You)

```
#id arrival runtime priority memory
1 4 1 10  129
2 4 0 1 197
3 4 0 2 181
4 7 1 10  82
5 9 4 2 9
```

As you can see, process 2 and 3 had runtime = 0 so they finished in the very same time clock that process 1 started, so actually process 1 didn't wait and started in its arriving clock cycle, this wait=0 & WTA=1.

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log

phase2 > code > exe > scheduler.perf

```
CPU utilization = 69.23%
Avg WTA = 0.60
Avg Waiting = 0.00
```

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log

phase2 > code > exe > memory.log

```
At time 4 allocated 197 bytes for process 2 from 129  to 325
At time 4 freed 197 bytes for process 2 from 129  to 325
At time 4 allocated 181 bytes for process 3 from 507  to 687
At time 4 freed 181 bytes for process 3 from 507  to 687
At time 4 allocated 129 bytes for process 1 from 817  to 945
At time 5 freed 129 bytes for process 1 from 817  to 945
At time 7 allocated 82  bytes for process 4 from 0  to 81
At time 8 freed 82  bytes for process 4 from 0  to 81
At time 9 allocated 9 bytes for process 5 from 82 to 90
At time 13  freed 9 bytes for process 5 from 82 to 90
```

RR (q=4)(Best Fit)

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log

phase2 > code > exe > procs.m

You, seconds ago | 1 author (You)

```
#id arrival runtime priority memory
1 4 2 3 13
2 5 0 7 22
3 5 2 9 159
4 8 2 2 156
5 10  1 9 174
6 11  6 1 208
7 11  2 0 141
```

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log

phase2 > code > exe > scheduler.log

```
At time 4 process 1 started arr 4 total 2 remain 2  wait 0
At time 6 process 1 finished arr 4  total 2 remain 0  wait 0  TA 2  WTA 1.00
At time 6 process 2 started arr 5 total 0 remain 0  wait 1
At time 6 process 2 finished arr 5  total 0 remain 0  wait 1  TA 1  WTA 1.00
At time 6 process 3 started arr 5 total 2 remain 2  wait 1
At time 8 process 3 finished arr 5  total 2 remain 0  wait 1  TA 3  WTA 1.50
At time 8 process 4 started arr 8 total 2 remain 2  wait 0
At time 10  process 4 finished arr 8  total 2 remain 0  wait 0  TA 2  WTA 1.00
At time 10  process 5 started arr 10  total 1 remain 1  wait 0
At time 11  process 5 finished arr 10 total 1 remain 0  wait 0  TA 1  WTA 1.00
At time 11  process 6 started arr 11  total 6 remain 6  wait 0
At time 15  process 6 stopped arr 11  total 6 remain 2  wait 0
At time 15  process 7 started arr 11  total 2 remain 2  wait 4
At time 17  process 7 finished arr 11 total 2 remain 0  wait 4  TA 6  WTA 3.00
At time 17  process 6 resumed arr 11  total 6 remain 2  wait 2
At time 19  process 6 finished arr 11 total 6 remain 0  wait 2  TA 8  WTA 1.33
```

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log

phase2 > code > exe > scheduler.perf

```
CPU utilization = 84.21%
Avg WTA = 1.40
Avg Waiting = 1.14
```

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log ×

phase2 > code > exe > memory.log

```
At time 4 allocated 13  bytes for process 1 from 0  to 12
At time 6 freed 13  bytes for process 1 from 0  to 12
At time 6 allocated 22  bytes for process 2 from 0  to 21
At time 6 freed 22  bytes for process 2 from 0  to 21
At time 6 allocated 159 bytes for process 3 from 0  to 158
At time 8 freed 159 bytes for process 3 from 0  to 158
At time 8 allocated 156 bytes for process 4 from 0  to 155
At time 10  freed 156 bytes for process 4 from 0  to 155
At time 10  allocated 174 bytes for process 5 from 0  to 173
At time 11  freed 174 bytes for process 5 from 0  to 173
At time 11  allocated 208 bytes for process 6 from 0  to 207
At time 15  allocated 141 bytes for process 7 from 208  to 348
At time 17  freed 141 bytes for process 7 from 208  to 348
At time 19  freed 208 bytes for process 6 from 0  to 207

```

HPF (Buddy System Allocation)

Makefile M | procs.m M × | scheduler.log | scheduler.perf

phase2 > code > exe > procs.m

You, seconds ago | 1 author (You)

```
#id arrival runtime priority memory
1 3 0 9 202
2 5 4 10  32
3 7 4 2 83
4 9 3 1 69
5 12  0 8 33

```

Makefile M | procs.m M | scheduler.log | scheduler.perf

phase2 > code > exe > scheduler.log

```
At time 3 process 1 started arr 3 total 0 remain 0  wait 0
At time 3 process 1 finished arr 3  total 0 remain 0  wait 0  TA 0  WTA 0.00
At time 5 process 2 started arr 5 total 4 remain 4  wait 0
At time 7 process 2 stopped arr 5 total 4 remain 2  wait 0
At time 7 process 3 started arr 7 total 4 remain 4  wait 0
At time 9 process 3 stopped arr 7 total 4 remain 2  wait 0
At time 9 process 4 started arr 9 total 3 remain 3  wait 0
At time 12  process 4 finished arr 9  total 3 remain 0  wait 0  TA 3  WTA 1.00
At time 12  process 3 resumed arr 7 total 4 remain 2  wait 3
At time 14  process 3 finished arr 7  total 4 remain 0  wait 3  TA 7  WTA 1.75
At time 14  process 5 started arr 12  total 0 remain 0  wait 2
At time 14  process 5 finished arr 12 total 0 remain 0  wait 2  TA 2  WTA 2.00
At time 14  process 2 resumed arr 5 total 4 remain 2  wait 7
At time 16  process 2 finished arr 5  total 4 remain 0  wait 7  TA 11 WTA 2.75
```

Makefile M | procs.m M | scheduler.log | scheduler.perf

phase2 > code > exe > scheduler.perf

```
CPU utilization = 81.25%
Avg WTA = 1.50
Avg Waiting = 2.40
```

Makefile M | procs.m M | scheduler.log | scheduler.perf | memory.log

phase2 > code > exe > memory.log

```
At time 3 allocated 256 bytes for process 1 from 0  to 255
At time 3 freed 256 bytes for process 1 from 0  to 255
At time 5 allocated 32  bytes for process 2 from 0  to 31
At time 7 allocated 128 bytes for process 3 from 128  to 255
At time 9 allocated 128 bytes for process 4 from 256  to 383
At time 12  freed 128 bytes for process 4 from 256  to 383
At time 14  freed 128 bytes for process 3 from 128  to 255
At time 14  allocated 64  bytes for process 5 from 64 to 127
At time 14  freed 64  bytes for process 5 from 64 to 127
At time 16  freed 32  bytes for process 2 from 0  to 31
```