

VIETNAM NATIONAL UNIVERSITY - HCM
Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering



OPERATING SYSTEM (LAB) (CO2018)

Assignment:

Simple Operating System

Instructor: Mr. Nguyễn Minh Trí
Students: Trần Quốc Anh - 1852247
Nguyễn Hoàng Dũng - 1852305

Ho Chi Minh City, June 2020



Contents

1	Introduction	2
1.1	Overview	2
1.2	Outcome	2
2	Scheduler	2
2.1	Question	2
2.2	Result - Gantt Diagram	2
3	Memory Management	3
3.1	Question	3
3.2	Result - RAM Status	3
4	Put It All Together	6



1 Introduction

1.1 Overview

The assignment is about simulating a simple operating system to help student understand the fundamental concepts of scheduling, synchronization and memory management. Figure 1 shows the overall architecture of the “operating system” we are going to implement. Generally, the OS has to manage two “virtual” resources: CPU(s) and RAM using two core components:

- Scheduler (and Dispatcher): determines with process is allowed to run on which CPU.
- Virtual memory engine (VME): isolates the memory space of each process from other. That is, although RAM is shared by multiple processes, each process do not know the existence of other. This is done by letting each process has its own virtual memory space and the Virtual memory engine will map and translate the virtual addresses provided by processes to corresponding physical addresses.

1.2 Outcome

After this assignment, student can understand partly the principle of a simple OS. They can draw the role and meaning of key modules in the OS as well as how it works.

2 Scheduler

2.1 Question

What is the advantage of using priority feedback queue in comparison with other scheduling algorithms you have learned?

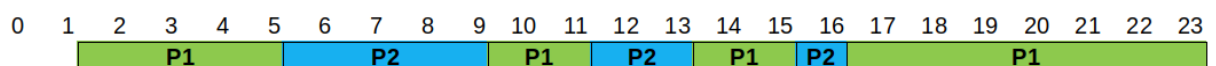
Answer: Priority Feedback Queue (PFQ) algorithm is designed based on “Multilevel Feedback Queue (MLFQ)” - allows a process to move between queues, with the style of "Round-Robin (RR)" and "Priority Scheduling" algorithm.

- Advantages of Priority Feedback Queue (PFQ) compared to other algorithms:

Unlike other algorithms, using PFQ allows processes to move back and forth between queues. If a process has a large CPU burst time, it will move to a lower-priority queue and vice versa, a process that waits too long in a low-priority queue will move into a higher-priority queue to reduce the indefinitely waiting status. Therefore, PFQ can avoid the process of having a long execution time of monopolizing the CPU (as in the case of FCFS) and avoiding the situation of processes with long execution time and waiting very long (as in the case of SJF, SRTF). PFQ also limited time slice issues and moved processes of Round-Robin.

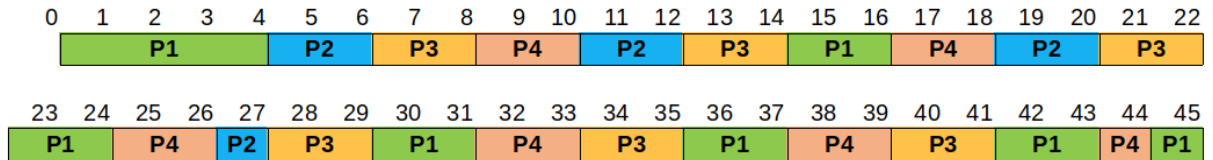
2.2 Result - Gantt Diagram

- Scheduling test 0 Gantt diagram





- Scheduling test 1 Gantt diagram



3 Memory Management

3.1 Question

What is the advantage and disadvantage of segmentation with paging?

Answer:

- Advantages:

- Paging reduces external fragmentation.
- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.

- Disadvantages:

- It may suffer from internal fragmentation.
- Page table requires extra memory space, so may not be good for a system having small RAM.
- Complex memory management algorithm
- Multi-level paging may lead to memory reference overhead.

3.2 Result - RAM Status

Below is the status of RAM after each memory allocation and deallocation function call. As can be seen, the number of pages used increases after each allocation call and equal to `num_pages`. Meanwhile it deletes the page with given address and all pages afterward (For example in *Memory management test 0*, when we deallocate index 000, all page from 000 to 013 (last page) is deleted, but these pages can also be re-allocated since after allocation 2 new pages, it uses page 001 and 002 instead of 016 and 017).

MEMORY MANAGEMENT TEST 0

```
1 -----ALLOCATION (size : 13535, num_pages : 14)-----
2 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
3 001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
4 002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
5 003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
6 004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
7 005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
8 006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
9 007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
```



```
10 008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
11 009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
12 010: 02800-02bfff - PID: 01 (idx 010, nxt: 011)
13 011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
14 012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
15 013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
16 -----ALLOCATION (size : 1568, num_pages : 2)-----
17 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
18 001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
19 002: 00800-00bfff - PID: 01 (idx 002, nxt: 003)
20 003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
21 004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
22 005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
23 006: 01800-01bfff - PID: 01 (idx 006, nxt: 007)
24 007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
25 008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
26 009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
27 010: 02800-02bfff - PID: 01 (idx 010, nxt: 011)
28 011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
29 012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
30 013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
31 014: 03800-03bfff - PID: 01 (idx 000, nxt: 015)
32 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
33 -----DEALLOCATION (index : 000)-----
34 014: 03800-03bfff - PID: 01 (idx 000, nxt: 015)
35 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
36 -----ALLOCATION (size : 1386, num_pages : 2)-----
37 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
38 001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
39 014: 03800-03bfff - PID: 01 (idx 000, nxt: 015)
40 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
41 -----ALLOCATION (size : 4564, num_pages : 5)-----
42 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
43 001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
44 002: 00800-00bfff - PID: 01 (idx 000, nxt: 003)
45 003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
46 004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
47 005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
48 006: 01800-01bfff - PID: 01 (idx 004, nxt: -01)
49 014: 03800-03bfff - PID: 01 (idx 000, nxt: 015)
50 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
```

MEMORY MANAGEMENT TEST 1

```
1 -----ALLOCATION (size : 13535, num_pages : 14)-----
2 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
3 001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
4 002: 00800-00bfff - PID: 01 (idx 002, nxt: 003)
5 003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
6 004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
7 005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
8 006: 01800-01bfff - PID: 01 (idx 006, nxt: 007)
9 007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
10 008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
11 009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
12 010: 02800-02bfff - PID: 01 (idx 010, nxt: 011)
13 011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
14 012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
15 013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
16 -----ALLOCATION (size : 1568, num_pages : 2)-----
17 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
```



```
18 001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
19 002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
20 003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
21 004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
22 005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
23 006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
24 007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
25 008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
26 009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
27 010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
28 011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
29 012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
30 013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
31 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
32 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
33 -----DEALLOCATION (index : 000)-----
34 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
35 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
36 -----ALLOCATION (size : 1386, num_pages : 2)-----
37 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
38 001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
39 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
40 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
41 -----ALLOCATION (size : 4564, num_pages : 5)-----
42 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
43 001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
44 002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
45 003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
46 004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
47 005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
48 006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
49 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
50 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
51 -----DEALLOCATION (index : 000)-----
52 002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
53 003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
54 004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
55 005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
56 006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
57 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
58 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
59 -----DEALLOCATION (index : 002)-----
60 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
61 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
62 -----DEALLOCATION (index : 014)-----
```

FINAL RESULT OF BOTH TEST

```
1 ----- MEMORY MANAGEMENT TEST 0 -----
2 ./mem input/proc/m0
3 000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
4 003e8: 15
5 001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
6 002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
7 003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
8 004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
9 005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
10 006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
11 014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
12 03814: 66
13 015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
```



```
14 NOTE: Read file output/m0 to verify your result
15 ----- MEMORY MANAGEMENT TEST 1 -----
16 ./mem input/proc/m1
17 NOTE: Read file output/m1 to verify your result (your implementation should print
    nothing)
```

4 Put It All Together

Since we are running multiple processes concurrently as well as the loader and the scheduler, the output every time we call make test is different. Below is one of the results we got.

```
1 ----- OS TEST 0 -----
2 ./os os_0
3 Time slot 0
4   Loaded a process at input/proc/p0, PID: 1
5 Time slot 1
6   CPU 1: Dispatched process 1
7 Time slot 2
8   Loaded a process at input/proc/p1, PID: 2
9 Time slot 3
10  CPU 0: Dispatched process 2
11  Loaded a process at input/proc/p1, PID: 3
12 Time slot 4
13  Loaded a process at input/proc/p1, PID: 4
14 Time slot 5
15 Time slot 6
16 Time slot 7
17  CPU 1: Put process 1 to run queue
18  CPU 1: Dispatched process 3
19 Time slot 8
20 Time slot 9
21  CPU 0: Put process 2 to run queue
22  CPU 0: Dispatched process 4
23 Time slot 10
24 Time slot 11
25 Time slot 12
26 Time slot 13
27  CPU 1: Put process 3 to run queue
28  CPU 1: Dispatched process 1
29 Time slot 14
30 Time slot 15
31  CPU 0: Put process 4 to run queue
32  CPU 0: Dispatched process 2
33 Time slot 16
34 Time slot 17
35  CPU 1: Processed 1 has finished
36  CPU 1: Dispatched process 3
37 Time slot 18
38 Time slot 19
39  CPU 0: Processed 2 has finished
40  CPU 0: Dispatched process 4
41 Time slot 20
42 Time slot 21
43  CPU 1: Processed 3 has finished
44  CPU 1 stopped
45 Time slot 22
46 Time slot 23
47  CPU 0: Processed 4 has finished
48  CPU 0 stopped
```



```
49 MEMORY CONTENT:
50 000: 00000-003ff - PID: 02 (idx 000, nxt: 001)
51 001: 00400-007ff - PID: 02 (idx 001, nxt: 007)
52 002: 00800-00bff - PID: 02 (idx 000, nxt: 003)
53 003: 00c00-00fff - PID: 02 (idx 001, nxt: 004)
54 004: 01000-013ff - PID: 02 (idx 002, nxt: 005)
55 005: 01400-017ff - PID: 02 (idx 003, nxt: -01)
56 006: 01800-01bff - PID: 03 (idx 000, nxt: 011)
57 007: 01c00-01fff - PID: 02 (idx 002, nxt: 008)
58 01de7: 0a
59 008: 02000-023ff - PID: 02 (idx 003, nxt: 009)
60 009: 02400-027ff - PID: 02 (idx 004, nxt: -01)
61 010: 02800-02bff - PID: 01 (idx 000, nxt: -01)
62 02814: 64
63 011: 02c00-02fff - PID: 03 (idx 001, nxt: 012)
64 012: 03000-033ff - PID: 03 (idx 002, nxt: 013)
65 013: 03400-037ff - PID: 03 (idx 003, nxt: -01)
66 014: 03800-03bff - PID: 04 (idx 000, nxt: 025)
67 015: 03c00-03fff - PID: 03 (idx 000, nxt: 016)
68 016: 04000-043ff - PID: 03 (idx 001, nxt: 017)
69 017: 04400-047ff - PID: 03 (idx 002, nxt: 018)
70 045e7: 0a
71 018: 04800-04bff - PID: 03 (idx 003, nxt: 019)
72 019: 04c00-04fff - PID: 03 (idx 004, nxt: -01)
73 020: 05000-053ff - PID: 04 (idx 000, nxt: 021)
74 021: 05400-057ff - PID: 04 (idx 001, nxt: 022)
75 022: 05800-05bff - PID: 04 (idx 002, nxt: 023)
76 059e7: 0a
77 023: 05c00-05fff - PID: 04 (idx 003, nxt: 024)
78 024: 06000-063ff - PID: 04 (idx 004, nxt: -01)
79 025: 06400-067ff - PID: 04 (idx 001, nxt: 026)
80 026: 06800-06bff - PID: 04 (idx 002, nxt: 027)
81 027: 06c00-06fff - PID: 04 (idx 003, nxt: -01)
82 NOTE: Read file output/os_0 to verify your result
83 ----- OS TEST 1 -----
84 ./os os_1
85 Time slot 0
86 Time slot 1
87   Loaded a process at input/proc/p0, PID: 1
88 Time slot 2
89   CPU 3: Dispatched process 1
90   Loaded a process at input/proc/s3, PID: 2
91 Time slot 3
92   CPU 2: Dispatched process 2
93 Time slot 4
94   CPU 3: Put process 1 to run queue
95   CPU 3: Dispatched process 1
96   Loaded a process at input/proc/m1, PID: 3
97 Time slot 5
98   CPU 2: Put process 2 to run queue
99   CPU 2: Dispatched process 2
100  CPU 1: Dispatched process 3
101  Loaded a process at input/proc/s2, PID: 4
102  CPU 3: Put process 1 to run queue
103  CPU 3: Dispatched process 4
104 Time slot 6
105  CPU 0: Dispatched process 1
106  Loaded a process at input/proc/m0, PID: 5
107  CPU 2: Put process 2 to run queue
108  CPU 2: Dispatched process 5
109 Time slot 7
110  CPU 1: Put process 3 to run queue
```




```
111 CPU 1: Dispatched process 2
112 CPU 3: Put process 4 to run queue
113 CPU 3: Dispatched process 3
114 Time slot 8
115 CPU 0: Put process 1 to run queue
116 CPU 0: Dispatched process 4
117 Loaded a process at input/proc/p1, PID: 6
118 CPU 2: Put process 5 to run queue
119 CPU 2: Dispatched process 1
120 CPU 1: Put process 2 to run queue
121 CPU 1: Dispatched process 6
122 Time slot 9
123 CPU 3: Put process 3 to run queue
124 CPU 3: Dispatched process 2
125 Time slot 10
126 CPU 0: Put process 4 to run queue
127 CPU 0: Dispatched process 5
128 Loaded a process at input/proc/s0, PID: 7
129 CPU 2: Put process 1 to run queue
130 CPU 2: Dispatched process 7
131 CPU 1: Put process 6 to run queue
132 CPU 1: Dispatched process 3
133 Time slot 11
134 CPU 3: Put process 2 to run queue
135 CPU 3: Dispatched process 4
136 Time slot 12
137 CPU 0: Put process 5 to run queue
138 CPU 0: Dispatched process 2
139 CPU 2: Put process 7 to run queue
140 CPU 2: Dispatched process 1
141 CPU 1: Put process 3 to run queue
142 CPU 1: Dispatched process 6
143 Time slot 13
144 CPU 3: Put process 4 to run queue
145 CPU 3: Dispatched process 4
146 CPU 0: Put process 2 to run queue
147 CPU 0: Dispatched process 7
148 Time slot 14
149 CPU 2: Processed 1 has finished
150 CPU 2: Dispatched process 5
151 CPU 1: Put process 6 to run queue
152 CPU 1: Dispatched process 3
153 Time slot 15
154 Loaded a process at input/proc/s1, PID: 8
155 CPU 3: Put process 4 to run queue
156 CPU 3: Dispatched process 8
157 CPU 0: Put process 7 to run queue
158 CPU 0: Dispatched process 4
159 Time slot 16
160 CPU 2: Put process 5 to run queue
161 CPU 2: Dispatched process 7
162 CPU 1: Processed 3 has finished
163 CPU 1: Dispatched process 2
164 Time slot 17
165 CPU 3: Put process 8 to run queue
166 CPU 3: Dispatched process 6
167 CPU 1: Processed 2 has finished
168 CPU 1: Dispatched process 8
169 CPU 0: Put process 4 to run queue
170 CPU 0: Dispatched process 5
171 Time slot 18
172 CPU 2: Put process 7 to run queue
```

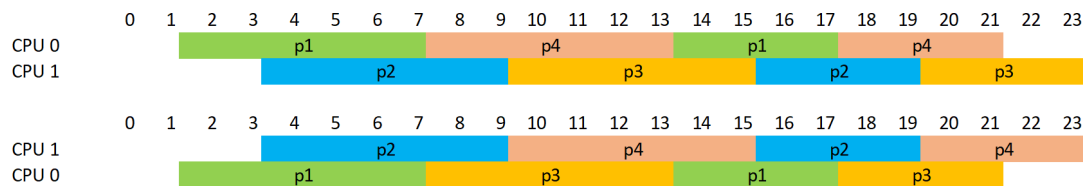


```
173 CPU 2: Dispatched process 4
174 CPU 0: Processed 5 has finished
175 CPU 0: Dispatched process 7
176 Time slot 19
177 CPU 3: Put process 6 to run queue
178 CPU 3: Dispatched process 6
179 CPU 1: Put process 8 to run queue
180 CPU 1: Dispatched process 8
181 Time slot 20
182 CPU 2: Processed 4 has finished
183 CPU 2 stopped
184 CPU 0: Put process 7 to run queue
185 CPU 0: Dispatched process 7
186 Time slot 21
187 CPU 3: Put process 6 to run queue
188 CPU 3: Dispatched process 6
189 CPU 1: Put process 8 to run queue
190 CPU 1: Dispatched process 8
191 Time slot 22
192 CPU 1: Processed 8 has finished
193 CPU 1 stopped
194 CPU 0: Put process 7 to run queue
195 CPU 0: Dispatched process 7
196 Time slot 23
197 CPU 3: Processed 6 has finished
198 CPU 3 stopped
199 Time slot 24
200 CPU 0: Put process 7 to run queue
201 CPU 0: Dispatched process 7
202 Time slot 25
203 Time slot 26
204 CPU 0: Put process 7 to run queue
205 CPU 0: Dispatched process 7
206 Time slot 27
207 CPU 0: Processed 7 has finished
208 CPU 0 stopped
209 MEMORY CONTENT:
210 000: 00000-003ff - PID: 05 (idx 000, nxt: 001)
211 00014: 66
212 001: 00400-007ff - PID: 05 (idx 001, nxt: -01)
213 002: 00800-00bff - PID: 05 (idx 000, nxt: 003)
214 003: 00c00-00fff - PID: 05 (idx 001, nxt: 004)
215 004: 01000-013ff - PID: 05 (idx 002, nxt: 005)
216 005: 01400-017ff - PID: 05 (idx 003, nxt: 006)
217 006: 01800-01bff - PID: 05 (idx 004, nxt: -01)
218 009: 02400-027ff - PID: 06 (idx 000, nxt: 010)
219 010: 02800-02bff - PID: 06 (idx 001, nxt: 011)
220 011: 02c00-02fff - PID: 06 (idx 002, nxt: 012)
221 02de7: 0a
222 012: 03000-033ff - PID: 06 (idx 003, nxt: 013)
223 013: 03400-037ff - PID: 06 (idx 004, nxt: -01)
224 016: 04000-043ff - PID: 06 (idx 000, nxt: 017)
225 017: 04400-047ff - PID: 06 (idx 001, nxt: 018)
226 018: 04800-04bff - PID: 06 (idx 002, nxt: 019)
227 019: 04c00-04fff - PID: 06 (idx 003, nxt: -01)
228 021: 05400-057ff - PID: 01 (idx 000, nxt: -01)
229 05414: 64
230 022: 05800-05bff - PID: 05 (idx 000, nxt: 023)
231 05be8: 15
232 023: 05c00-05fff - PID: 05 (idx 001, nxt: -01)
233 NOTE: Read file output/os_1 to verify your result
```



To verify the result compare to file *output/os_0*, we need methods to check both the scheduler and memory management works correctly.

For the scheduler, we draw 2 Gantt charts of each test to observe if each process has the same execution time. Below are 2 Gantt charts for Test 0, the upper chart is collected from *output/os_0* and the lower chart is from our output.



Also, since p1, p2 and p3 have the same priority, the result can be changed due to our comparison (whether we dispatch the first or the last process with the highest priority in the ready_queue). Different scheduler result also leads to different memory allocation result. The result in *output/os_0* file is

```
1 MEMORY CONTENT:
2 000: 00000-003ff - PID: 04 (idx 000, nxt: 001)
3 001: 00400-007ff - PID: 04 (idx 001, nxt: 002)
4 002: 00800-00bff - PID: 04 (idx 002, nxt: 003)
5 003: 00c00-00fff - PID: 04 (idx 003, nxt: -01)
6 004: 01000-013ff - PID: 03 (idx 000, nxt: 005)
7 005: 01400-017ff - PID: 03 (idx 001, nxt: 006)
8 006: 01800-01bff - PID: 03 (idx 002, nxt: 012)
9 007: 01c00-01fff - PID: 02 (idx 000, nxt: 008)
10 008: 02000-023ff - PID: 02 (idx 001, nxt: 009)
11 009: 02400-027ff - PID: 02 (idx 002, nxt: 010)
12 025e7: 0a
13 010: 02800-02bff - PID: 02 (idx 003, nxt: 011)
14 011: 02c00-02fff - PID: 02 (idx 004, nxt: -01)
15 012: 03000-033ff - PID: 03 (idx 003, nxt: -01)
16 014: 03800-03bff - PID: 04 (idx 000, nxt: 015)
17 015: 03c00-03fff - PID: 04 (idx 001, nxt: 016)
18 016: 04000-043ff - PID: 04 (idx 002, nxt: 017)
19 041e7: 0a
20 017: 04400-047ff - PID: 04 (idx 003, nxt: 018)
21 018: 04800-04bff - PID: 04 (idx 004, nxt: -01)
22 023: 05c00-05fff - PID: 02 (idx 000, nxt: 024)
23 024: 06000-063ff - PID: 02 (idx 001, nxt: 025)
24 025: 06400-067ff - PID: 02 (idx 002, nxt: 026)
25 026: 06800-06bff - PID: 02 (idx 003, nxt: -01)
26 047: 0bc00-0bfff - PID: 01 (idx 000, nxt: -01)
27 0bc14: 64
28 057: 0e400-0e7ff - PID: 03 (idx 000, nxt: 058)
29 058: 0e800-0ebff - PID: 03 (idx 001, nxt: 059)
30 059: 0ec00-0efff - PID: 03 (idx 002, nxt: 060)
31 0ede7: 0a
32 060: 0f000-0f3ff - PID: 03 (idx 003, nxt: 061)
33 061: 0f400-0f7ff - PID: 03 (idx 004, nxt: -01)
```



In order to verify the result, we need to check the map of memory if it is identical or not.

In *output/os_0*, the map is :

- 000->001->002->003. PID: 04
- 004->005->006->012. PID: 03
- 007->008->009->010->011. PID: 02
- 014->015->016->017->018. PID: 04
- 023->024->025->026. PID: 02
- 047. PID: 01
- 057->058->059->060->061. PID: 03

In our output file, the map is :

- 000->001->007->008->009. PID: 02
- 002->003->004->005. PID: 02
- 006->011->012->013. PID: 03
- 010. PID: 01
- 014->025->026->027. PID: 04
- 015->016->017->018->019. PID: 03
- 020->021->022->023->024. PID: 04

We have the same number of pages create within the same process. Therefore, we can consider the result is true.

The same explanation can be used for Test 1 below.

