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

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

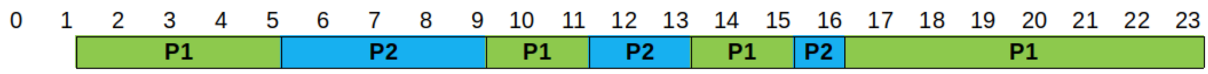
What is the advantage of using priority feedback queue in comparison with other scheduling algorithms you have learned?

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.

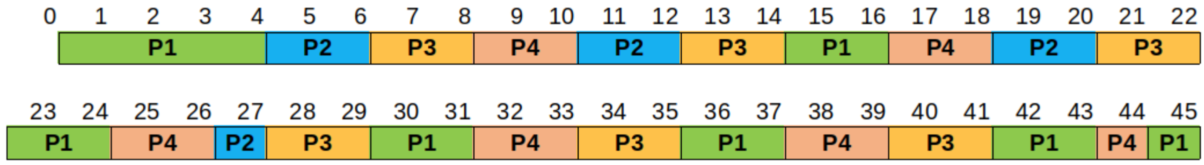
- Advantages of Priority Feedback Queue (PFQ) compared to other algorithms:
Unlike other algorithms, using PFQ allows processes to move back and forth between queues. If a process has a large CPU burst time, it will move to a lower-priority queue and vice versa, a process that waits too long in a low-priority 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 suffers 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

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

```
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
-----ALLOCATION (size : 1568, num_pages : 2)-----
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----DEALLOCATION (index : 000)-----
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----ALLOCATION (size : 1386, num_pages : 2)-----
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----ALLOCATION (size : 4564, num_pages : 5)-----
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
```

MEMORY MANAGEMENT TEST 1

```
-----ALLOCATION (size : 13535, num_pages : 14)-----
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
-----ALLOCATION (size : 1568, num_pages : 2)-----
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
```

```
001: 00400-007ff - PID: 01 (idx 001, nxt: 002)
002: 00800-00bff - PID: 01 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 003, nxt: 004)
004: 01000-013ff - PID: 01 (idx 004, nxt: 005)
005: 01400-017ff - PID: 01 (idx 005, nxt: 006)
006: 01800-01bff - PID: 01 (idx 006, nxt: 007)
007: 01c00-01fff - PID: 01 (idx 007, nxt: 008)
008: 02000-023ff - PID: 01 (idx 008, nxt: 009)
009: 02400-027ff - PID: 01 (idx 009, nxt: 010)
010: 02800-02bff - PID: 01 (idx 010, nxt: 011)
011: 02c00-02fff - PID: 01 (idx 011, nxt: 012)
012: 03000-033ff - PID: 01 (idx 012, nxt: 013)
013: 03400-037ff - PID: 01 (idx 013, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----DEALLOCATION (index : 000)-----
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----ALLOCATION (size : 1386, num_pages : 2)-----
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----ALLOCATION (size : 4564, num_pages : 5)-----
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----DEALLOCATION (index : 000)-----
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----DEALLOCATION (index : 002)-----
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
-----DEALLOCATION (index : 014)-----
```

FINAL RESULT OF BOTH TEST

```
------ MEMORY MANAGEMENT TEST 0 -------------------------------------
./mem input/proc/m0
000: 00000-003ff - PID: 01 (idx 000, nxt: 001)
  003e8: 15
001: 00400-007ff - PID: 01 (idx 001, nxt: -01)
002: 00800-00bff - PID: 01 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 01 (idx 001, nxt: 004)
004: 01000-013ff - PID: 01 (idx 002, nxt: 005)
005: 01400-017ff - PID: 01 (idx 003, nxt: 006)
006: 01800-01bff - PID: 01 (idx 004, nxt: -01)
014: 03800-03bff - PID: 01 (idx 000, nxt: 015)
  03814: 66
015: 03c00-03fff - PID: 01 (idx 001, nxt: -01)
```

```
NOTE: Read file output/m0 to verify your result
------ MEMORY MANAGEMENT TEST 1 ------------------------------------
./mem input/proc/m1
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.

```
----- OS TEST 0 ----------------------------------------------------
./os os_0
Time slot   0
    Loaded a process at input/proc/p0, PID: 1
Time slot   1
  CPU 1: Dispatched process  1
Time slot   2
  Loaded a process at input/proc/p1, PID: 2
Time slot   3
  CPU 0: Dispatched process  2
  Loaded a process at input/proc/p1, PID: 3
Time slot   4
  Loaded a process at input/proc/p1, PID: 4
Time slot   5
Time slot   6
Time slot   7
  CPU 1: Put process  1 to run queue
  CPU 1: Dispatched process  3
Time slot   8
Time slot   9
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  4
Time slot  10
Time slot  11
Time slot  12
Time slot  13
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  1
Time slot  14
Time slot  15
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  2
Time slot  16
Time slot  17
  CPU 1: Processed  1 has finished
  CPU 1: Dispatched process  3
Time slot  18
Time slot  19
  CPU 0: Processed  2 has finished
  CPU 0: Dispatched process  4
Time slot  20
Time slot  21
  CPU 1: Processed  3 has finished
  CPU 1 stopped
Time slot  22
Time slot  23
  CPU 0: Processed  4 has finished
  CPU 0 stopped
```

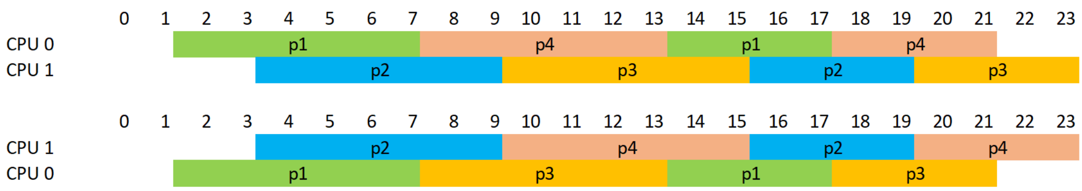
```
MEMORY CONTENT:
000: 00000-003ff - PID: 02 (idx 000, nxt: 001)
001: 00400-007ff - PID: 02 (idx 001, nxt: 007)
002: 00800-00bff - PID: 02 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 02 (idx 001, nxt: 004)
004: 01000-013ff - PID: 02 (idx 002, nxt: 005)
005: 01400-017ff - PID: 02 (idx 003, nxt: -01)
006: 01800-01bff - PID: 03 (idx 000, nxt: 011)
007: 01c00-01fff - PID: 02 (idx 002, nxt: 008)
  01de7: 0a
008: 02000-023ff - PID: 02 (idx 003, nxt: 009)
009: 02400-027ff - PID: 02 (idx 004, nxt: -01)
010: 02800-02bff - PID: 01 (idx 000, nxt: -01)
  02814: 64
011: 02c00-02fff - PID: 03 (idx 001, nxt: 012)
012: 03000-033ff - PID: 03 (idx 002, nxt: 013)
013: 03400-037ff - PID: 03 (idx 003, nxt: -01)
014: 03800-03bff - PID: 04 (idx 000, nxt: 025)
015: 03c00-03fff - PID: 03 (idx 000, nxt: 016)
016: 04000-043ff - PID: 03 (idx 001, nxt: 017)
017: 04400-047ff - PID: 03 (idx 002, nxt: 018)
  045e7: 0a
018: 04800-04bff - PID: 03 (idx 003, nxt: 019)
019: 04c00-04fff - PID: 03 (idx 004, nxt: -01)
020: 05000-053ff - PID: 04 (idx 000, nxt: 021)
021: 05400-057ff - PID: 04 (idx 001, nxt: 022)
022: 05800-05bff - PID: 04 (idx 002, nxt: 023)
  059e7: 0a
023: 05c00-05fff - PID: 04 (idx 003, nxt: 024)
024: 06000-063ff - PID: 04 (idx 004, nxt: -01)
025: 06400-067ff - PID: 04 (idx 001, nxt: 026)
026: 06800-06bff - PID: 04 (idx 002, nxt: 027)
027: 06c00-06fff - PID: 04 (idx 003, nxt: -01)
NOTE: Read file output/os_0 to verify your result
----- OS TEST 1 ----------------------------------------------------
./os os_1
Time slot   0
Time slot   1
  Loaded a process at input/proc/p0, PID: 1
Time slot   2
  CPU 3: Dispatched process  1
  Loaded a process at input/proc/s3, PID: 2
Time slot   3
  CPU 2: Dispatched process  2
Time slot   4
  CPU 3: Put process  1 to run queue
  CPU 3: Dispatched process  1
  Loaded a process at input/proc/m1, PID: 3
Time slot   5
  CPU 2: Put process  2 to run queue
  CPU 2: Dispatched process  2
  CPU 1: Dispatched process  3
  Loaded a process at input/proc/s2, PID: 4
  CPU 3: Put process  1 to run queue
  CPU 3: Dispatched process  4
Time slot   6
  CPU 0: Dispatched process  1
  Loaded a process at input/proc/m0, PID: 5
  CPU 2: Put process  2 to run queue
  CPU 2: Dispatched process  5
Time slot   7
  CPU 1: Put process  3 to run queue
```

```
  CPU 1: Dispatched process  2
  CPU 3: Put process  4 to run queue
  CPU 3: Dispatched process  3
Time slot   8
  CPU 0: Put process  1 to run queue
  CPU 0: Dispatched process  4
  Loaded a process at input/proc/p1, PID: 6
  CPU 2: Put process  5 to run queue
  CPU 2: Dispatched process  1
  CPU 1: Put process  2 to run queue
  CPU 1: Dispatched process  6
Time slot   9
  CPU 3: Put process  3 to run queue
  CPU 3: Dispatched process  2
Time slot  10
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  5
  Loaded a process at input/proc/s0, PID: 7
  CPU 2: Put process  1 to run queue
  CPU 2: Dispatched process  7
  CPU 1: Put process  6 to run queue
  CPU 1: Dispatched process  3
Time slot  11
  CPU 3: Put process  2 to run queue
  CPU 3: Dispatched process  4
Time slot  12
  CPU 0: Put process  5 to run queue
  CPU 0: Dispatched process  2
  CPU 2: Put process  7 to run queue
  CPU 2: Dispatched process  1
  CPU 1: Put process  3 to run queue
  CPU 1: Dispatched process  6
Time slot  13
  CPU 3: Put process  4 to run queue
  CPU 3: Dispatched process  4
  CPU 0: Put process  2 to run queue
  CPU 0: Dispatched process  7
Time slot  14
  CPU 2: Processed  1 has finished
  CPU 2: Dispatched process  5
  CPU 1: Put process  6 to run queue
  CPU 1: Dispatched process  3
Time slot  15
  Loaded a process at input/proc/s1, PID: 8
  CPU 3: Put process  4 to run queue
  CPU 3: Dispatched process  8
  CPU 0: Put process  7 to run queue
  CPU 0: Dispatched process  4
Time slot  16
  CPU 2: Put process  5 to run queue
  CPU 2: Dispatched process  7
  CPU 1: Processed  3 has finished
  CPU 1: Dispatched process  2
Time slot  17
  CPU 3: Put process  8 to run queue
  CPU 3: Dispatched process  6
  CPU 1: Processed  2 has finished
  CPU 1: Dispatched process  8
  CPU 0: Put process  4 to run queue
  CPU 0: Dispatched process  5
Time slot  18
  CPU 2: Put process  7 to run queue
```

```
  CPU 2: Dispatched process  4
  CPU 0: Processed  5 has finished
  CPU 0: Dispatched process  7
Time slot  19
  CPU 3: Put process  6 to run queue
  CPU 3: Dispatched process  6
  CPU 1: Put process  8 to run queue
  CPU 1: Dispatched process  8
Time slot  20
  CPU 2: Processed  4 has finished
  CPU 2 stopped
  CPU 0: Put process  7 to run queue
  CPU 0: Dispatched process  7
Time slot  21
  CPU 3: Put process  6 to run queue
  CPU 3: Dispatched process  6
  CPU 1: Put process  8 to run queue
  CPU 1: Dispatched process  8
Time slot  22
  CPU 1: Processed  8 has finished
  CPU 1 stopped
  CPU 0: Put process  7 to run queue
  CPU 0: Dispatched process  7
Time slot  23
  CPU 3: Processed  6 has finished
  CPU 3 stopped
Time slot  24
  CPU 0: Put process  7 to run queue
  CPU 0: Dispatched process  7
Time slot  25
Time slot  26
  CPU 0: Put process  7 to run queue
  CPU 0: Dispatched process  7
Time slot  27
  CPU 0: Processed  7 has finished
  CPU 0 stopped
MEMORY CONTENT:
000: 00000-003ff - PID: 05 (idx 000, nxt: 001)
  00014: 66
001: 00400-007ff - PID: 05 (idx 001, nxt: -01)
002: 00800-00bff - PID: 05 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 05 (idx 001, nxt: 004)
004: 01000-013ff - PID: 05 (idx 002, nxt: 005)
005: 01400-017ff - PID: 05 (idx 003, nxt: 006)
006: 01800-01bff - PID: 05 (idx 004, nxt: -01)
009: 02400-027ff - PID: 06 (idx 000, nxt: 010)
010: 02800-02bff - PID: 06 (idx 001, nxt: 011)
011: 02c00-02fff - PID: 06 (idx 002, nxt: 012)
  02de7: 0a
012: 03000-033ff - PID: 06 (idx 003, nxt: 013)
013: 03400-037ff - PID: 06 (idx 004, nxt: -01)
016: 04000-043ff - PID: 06 (idx 000, nxt: 017)
017: 04400-047ff - PID: 06 (idx 001, nxt: 018)
018: 04800-04bff - PID: 06 (idx 002, nxt: 019)
019: 04c00-04fff - PID: 06 (idx 003, nxt: -01)
021: 05400-057ff - PID: 01 (idx 000, nxt: -01)
  05414: 64
022: 05800-05bff - PID: 05 (idx 000, nxt: 023)
  05be8: 15
023: 05c00-05fff - PID: 05 (idx 001, nxt: -01)
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 draws 2 Gantt charts of each test to observe if each process has the same execution time. Below is 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 comparision (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

```
MEMORY CONTENT:
000: 00000-003ff - PID: 04 (idx 000, nxt: 001)
001: 00400-007ff - PID: 04 (idx 001, nxt: 002)
002: 00800-00bff - PID: 04 (idx 002, nxt: 003)
003: 00c00-00fff - PID: 04 (idx 003, nxt: -01)
004: 01000-013ff - PID: 03 (idx 000, nxt: 005)
005: 01400-017ff - PID: 03 (idx 001, nxt: 006)
006: 01800-01bff - PID: 03 (idx 002, nxt: 012)
007: 01c00-01fff - PID: 02 (idx 000, nxt: 008)
008: 02000-023ff - PID: 02 (idx 001, nxt: 009)
009: 02400-027ff - PID: 02 (idx 002, nxt: 010)
  025e7: 0a
010: 02800-02bff - PID: 02 (idx 003, nxt: 011)
011: 02c00-02fff - PID: 02 (idx 004, nxt: -01)
012: 03000-033ff - PID: 03 (idx 003, nxt: -01)
014: 03800-03bff - PID: 04 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 04 (idx 001, nxt: 016)
016: 04000-043ff - PID: 04 (idx 002, nxt: 017)
  041e7: 0a
017: 04400-047ff - PID: 04 (idx 003, nxt: 018)
018: 04800-04bff - PID: 04 (idx 004, nxt: -01)
023: 05c00-05fff - PID: 02 (idx 000, nxt: 024)
024: 06000-063ff - PID: 02 (idx 001, nxt: 025)
025: 06400-067ff - PID: 02 (idx 002, nxt: 026)
026: 06800-06bff - PID: 02 (idx 003, nxt: -01)
047: 0bc00-0bfff - PID: 01 (idx 000, nxt: -01)
  0bc14: 64
057: 0e400-0e7ff - PID: 03 (idx 000, nxt: 058)
058: 0e800-0ebff - PID: 03 (idx 001, nxt: 059)
059: 0ec00-0efff - PID: 03 (idx 002, nxt: 060)
  0ede7: 0a
060: 0f000-0f3ff - PID: 03 (idx 003, nxt: 061)
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.