ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Operating System (CO2017)

Assignment 2 Report:

SIMPLE OPERATING SYSTEM

Giảng viên:	Trần Việt Toản	
Sinh viên:	Trần Quốc Việt	MSSV : 1915919
	Nguyễn Việt Đức	MSSV : 1913167
	Nguyễn Thế Duy	MSSV : 1912912
	Trương Ngọc Trung Anh	MSSV : 2020004

TP. HỒ CHÍ MINH, THÁNG 5/2021

Mục lục

MỤC LỤC

1 Scheduling

Hiện thực *enqueue()* (trong *queue.c*) : thêm một process (đại diện bởi PCB) vào hàng đợi và tăng kích thước hàng đợi.

```c
void enqueue(struct queue_t * q, struct pcb_t * proc) {
    /* add new pcb to queue [q] if queue is not full */
    if (q->size != MAX_QUEUE_SIZE)
        q->proc[q->size++] = proc;
}
```

Hiện thực *dequeue()* (trong *queue.c*):

- Kiểm tra hàng đợi, nếu hàng đợi rỗng ta trả về NULL.
- Duyệt hết process, tìm process có độ ưu tiên cao nhất, lưu lại địa chỉ của process. Nếu có 2 process có cùng độ ưu tiên, ta ưu tiên lấy process đầu hàng đợi.
- Đem process ở cuối hàng đợi gán vào vị trí process vừa lấy ra.

```c
struct pcb_t * dequeue(struct queue_t * q) {
    /* pop the pcb with highest prioprity from queue [q] */
    if (q->size == 0) return NULL;

    int temp = 0;
    for (int i = 1; i < q->size; i++) {
        if (q->proc[i]->priority > q->proc[temp]->priority) {
            temp = i;
        }
    }
    struct pcb_t * res = q->proc[temp];
    q->proc[temp] = q->proc[--q->size];

    return res;
}
```

Hiện thực *get_proc()* (trong *sched.c*):

- Lấy process có độ ưu tiên cao nhất từ *ready_queue* để định thời, nếu *ready_queue* rỗng thì chuyển tất cả process từ *run_queue* sang *ready_queue*.
- Trả về PCB đã lấy được.

```
1  struct pcb_t * get_proc(void) {
2      struct pcb_t * proc = NULL;
3      /*TODO: get a process from [ready_queue]. If ready queue
4       * is empty, push all processes in [run_queue] back to
5       * [ready_queue] and return the highest priority one.
6       * Remember to use lock to protect the queue.
7       * */
8      pthread_mutex_lock(&queue_lock);
9      if (empty(&ready_queue))
10         while(!empty(&run_queue))
11             ready_queue.proc[ready_queue.size++] = run_queue.proc[--run_queue.size];
12
13     if (!empty(&ready_queue))
14         proc = dequeue(&ready_queue);
15     pthread_mutex_unlock(&queue_lock);
16
17     return proc;
18 }
```

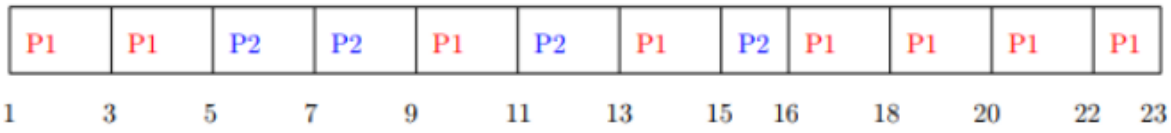
1.1 Scheduling test 0

- Kết quả của *sched_0*:

```
duc@duc:~/Assignment_2/source_code$ ./os sched_0
Time slot   0
        Loaded a process at input/proc/s0, PID: 1
Time slot   1
        CPU 0: Dispatched process  1
Time slot   2
Time slot   3
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot   4
        Loaded a process at input/proc/s1, PID: 2
Time slot   5
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  2
Time slot   6
Time slot   7
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  2
Time slot   8
Time slot   9
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  1
Time slot  10
Time slot  11
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  2
```

```
Time slot   12
Time slot   13
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  1
Time slot   14
Time slot   15
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  2
Time slot   16
        CPU 0: Processed  2 has finished
        CPU 0: Dispatched process  1
Time slot   17
Time slot   18
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot   19
Time slot   20
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot   21
Time slot   22
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot   23
        CPU 0: Processed  1 has finished
        CPU 0 stopped
```

- Giản đồ Gantt:



Trong đó, quantum time = 2 time slot:

- Trong khoảng 3-5 có diễn ra việc load process 2.
- Khoảng 7-9 do *ready_ queue* trống nên sẽ chuyển các process từ *run_ queue* về lại *ready_ queue* vả đưa ra process có độ ưu tiên cao hơn ở đây ta thấy P2 được chọn tương tự với các phần còn lại.
- Khoảng 15-16 P2 kết thúc, thời gian còn lại để chạy P1.

1.2 Scheduling test 1

- Kết quả của *sched_1*:

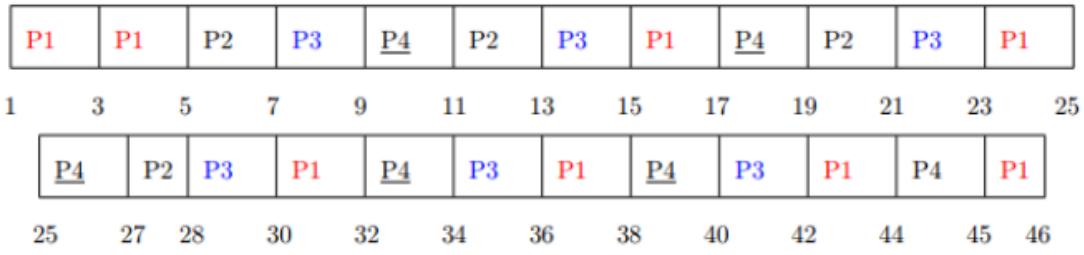
```
duc@duc:~/Assignment_2/source_code$ ./os sched_1
Time slot   0
        Loaded a process at input/proc/s0, PID: 1
Time slot   1
        CPU 0: Dispatched process  1
Time slot   2
Time slot   3
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot   4
        Loaded a process at input/proc/s1, PID: 2
Time slot   5
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  2
Time slot   6
        Loaded a process at input/proc/s2, PID: 3
Time slot   7
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  3
        Loaded a process at input/proc/s3, PID: 4
```

```
Time slot   8
Time slot   9
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  4
Time slot  10
Time slot  11
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  3
Time slot  12
Time slot  13
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  2
Time slot  14
Time slot  15
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  1
Time slot  16
Time slot  17
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot  18
Time slot  19
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  2
```

```
Time slot  20
Time slot  21
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  3
Time slot  22
Time slot  23
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  1
Time slot  24
Time slot  25
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot  26
Time slot  27
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  3
Time slot  28
Time slot  29
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  2
Time slot  30
        CPU 0: Processed  2 has finished
        CPU 0: Dispatched process  1
Time slot  31
Time slot  32
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
```

```
Time slot   33
Time slot   34
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  3
Time slot   35
Time slot   36
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  1
Time slot   37
Time slot   38
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot   39
Time slot   40
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  3
Time slot   41
Time slot   42
        CPU 0: Processed  3 has finished
        CPU 0: Dispatched process  1
Time slot   43
Time slot   44
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot   45
        CPU 0: Processed  4 has finished
        CPU 0: Dispatched process  1
Time slot   46
        CPU 0: Processed  1 has finished
        CPU 0 stopped
```

- Giản đồ Gantt:



- Quantum time = 2 time slot.

- Tương tự như trên, mỗi khi *ready_ queue* trống hệ thống tự động chuyển tất cả các process từ *run_ queue* về và đưa ra process có độ ưu tiên cao nhất để chạy:

- P2 kết thúc trong khoảng 29-30.
- P3 kết thúc trong khoảng 40-42.
- P4 kết thúc trong khoảng 44-45.
- P1 kết thúc trong khoảng 45-46.

1.3 Question

Question: What is the advantage of using prority feedback queue in comparision with other scheduling algorithms you have learned?

Answer:

- Giải thuật này giúp giảm thiểu thời gian phản hồi (response time).
- Thuật toán lập lịch này cho phép các quy trình khác nhau di chuyển giữa các hàng đợi khác nhau.
- Như chúng ta biết rằng thời gian chạy của các tiến trình không được biết trước. Lập lịch này chủ yếu chạy trong một khoảng thời gian nhất định (quantum time) và sau đó nó có thể thay đổi mức độ ưu tiên của quá trình nếu quá trình kéo dài. Do đó, thuật toán lập lịch này chủ yếu học từ hành vi trong quá khứ của các quy trình và sau đó nó có thể dự đoán hành vi trong tương lai của các quy trình. Theo cách này, giải thuật sẽ cố gắng chạy một quy trình ngắn hơn trước tiên, điều này đổi lại dẫn đến việc tối ưu hóa thời gian quay vòng (turn around time).

2 Memory Management

Địa chỉ ảo gồm 20 bit, 5 bit đầu là địa chỉ trong bảng phân đoạn, 5 bit tiếp theo là địa chỉ trong bảng phân trang, 10 bit cuối cùng chính là offset.
Hiện thực *get_ page_ table()* (trong *mem.c*). Tìm kiếm địa chi của bảng phân trang từ bảng phân đoạn: ta duyệt hết bảng phân đoạn để tìm bảng phân trang, nếu không tìm được trả về NULL.

```c
/* Search for page table table from the a segment table */
static struct page_table_t *get_page_table(
    addr_t index, // segment index
    struct seg_table_t *seg_table)
{ // first level table

    /*
     * TODO: Given the Segment index [index], you must go through each
     * row of the segment table [seg_table] and check if the v_index
     * field of the row is equal to the index
     *
     * */

    if (!seg_table)
        return NULL;

    for (int i = 0; i < seg_table->size; i++)
        if (index == seg_table->table[i].v_index)
            return seg_table->table[i].pages;

    return NULL;
}
```

Hiện thực *translate* (trong *mem.c*): Đổi từ địa chỉ ảo sang địa chỉ vật lí.

- Tách địa chỉ ảo thành 3 phần: *first_lv* (5 bit đầu), *second_lv* (5 bit sau đó) và *offset* (10 bit cuối).
- Dùng hàm *get_page_table* để tìm bảng phân trang từ *first_lv* nếu kết trả về NULL, ta trả giá trị hàm *translate* về 0 – việc chuyển đổi không thành công.
- Từ bảng phân trang, tìm kếm index bằng với *second_lv*, nếu không tìm thấy, trả giá trị hàm *translate* về 0 – việc chuyển đổi không thành công.
- Khi tìm được index trong bảng phân trang, ta tìm được địa chỉ vật lý bằng cách kết hợp địa chỉ trong bảng phân trang và offset của địa chỉ ảo.

```c
static int translate(
    addr_t virtual_addr,    // Given virtual address
    addr_t *physical_addr, // Physical address to be returned
    struct pcb_t *proc) {  // Process uses given virtual address

    /* Offset of the virtual address */
    addr_t offset = get_offset(virtual_addr);
    /* The first layer index */
    addr_t first_lv = get_first_lv(virtual_addr);
    /* The second layer index */
    addr_t second_lv = get_second_lv(virtual_addr);

    /* Search in the first level */
    struct page_table_t *page_table = get_page_table(first_lv, proc->seg_table);
    if (!page_table) return 0;

    for (int i = 0; i < page_table->size; i++) {
        if (page_table->table[i].v_index == second_lv) {
            /* TODO: Concatenate the offset of the virtual addess
             * to [p_index] field of page_table->table[i] to
             * produce the correct physical address and save it to
             * [*physical_addr]  */

            *physical_addr = (page_table->table[i].p_index << OFFSET_LEN) | (offset);
            return 1;
        }
    }
    return 0;
}
```

2.1 Memory Management test 0

- Trạng thái của bộ nhớ:

Trong đó lần lượt là index của page, địa chỉ bắt đầu và kết thúc của page trong physical memory, pid của process, index của page trong số các page được cấp phát cho process và index của page kế tiếp trong process đó

```
duc@duc:~/Assignment_2/source_code$ ./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)
```

2.2 Memory Management test 1

Nội dung của m1: 1 8

alloc 13535 0

alloc 1568 1

free 0

alloc 1386 2

alloc 4564 4

free 2

free 4

free 1

Vì sau khi cấp phát bộ nhớ và lưu địa chỉ của byte đầu tiên của mỗi vùng nhớ được cấp phát vào thanh ghi 0, 1, 2, 4 sau đó ta lại dùng lệnh free 0, free 1, free 2, free 4 nên sau khi thực thi chương trình này nội dung của RAM là trống, do đó không có gì được in ra console.

```
duc@duc:~/Assignment_2/source_code$ ./mem ./input/proc/m1
duc@duc:~/Assignment_2/source_code$
```

2.3 Question

Question: What are the advantages and disadvantages of segmentation with paging?

Answer:

- **Ưu điểm:**
  - Tiết kiệm bộ nhớ so với chỉ phân trang. Kích thước bảng phân trang được giới hạn bởi kích thước segment, bảng phân đoạn chỉ có một entry trên mỗi segment.
  - Chia sẻ được từng trang riêng biệt như Paging.
  - Chia sẻ được toàn bộ segment bằng việc chia sẻ entry trong bảng phân đoạn của segment đó, hoặc chia sẻ cả bản phân trang của nó.
  - Giữ được hầu hết ưu điểm của phân trang: đơn giản việc cấp phát bộ nhớ, loại bỏ phân mảnh ngoại
  - Giải quyết vấn đề phân mảnh ngoại của giải thuật phân đoạn bằng cách phân trang trong mỗi word.
- **Nhược điểm:**
  - Còn tồn tại phân mảnh nội.

3 Overall

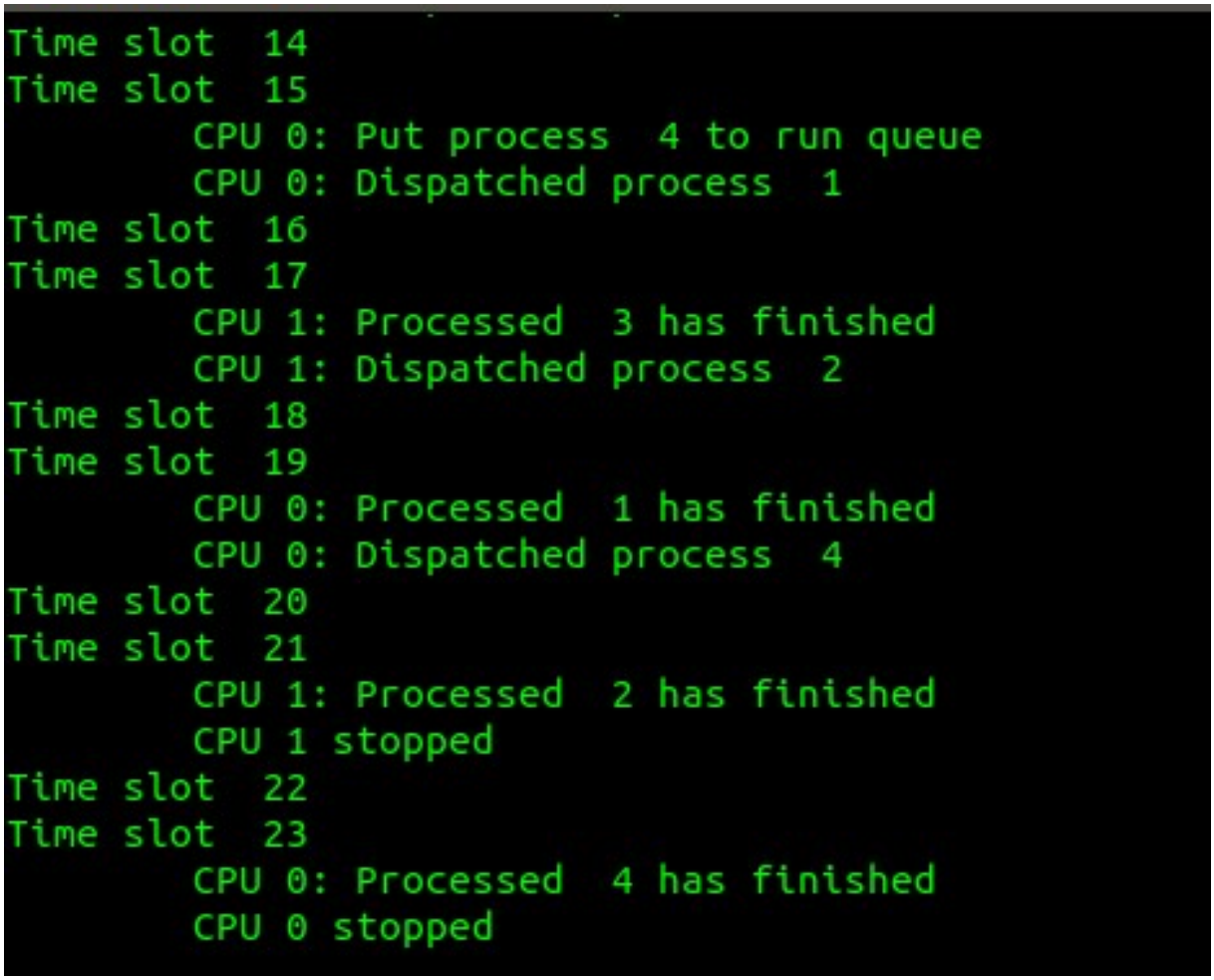
Trong bài tập lớn này, chúng ta đã mô phỏng được một hệ điều hành đơn giản gồm có 3 yếu tố chính cấu tạo nên là:

- **Scheduler:** giúp chúng ta xác định được process nào được sử dụng CPU tại một thời điểm bằng cách sử dụng giải thuật priority feedback queue. Cách hoạt động cụ thể của việc này thì nhóm em đã trình bày ở mục 1.2.
- **Memory management:** Cung cấp một vùng nhớ cho mỗi process bằng kỹ thuật phân đoạn kết hợp với phân trang (giải thích rõ ở mục 2.2).
- **Synchronization:** Bởi vì hệ điều hành chạy với nhiều CPU, nên các dữ liệu chia sẻ có thể được truy cập cùng lúc bởi nhiều process khác nhau. Do đó, chúng ta phải sử dụng lock để bảo vệ những dữ liệu chia sẻ đó (queue, ram).

3.1 OS test 0

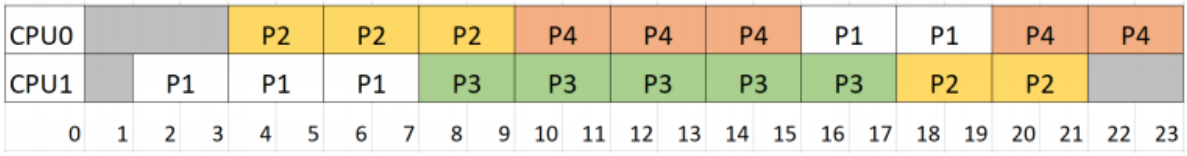
- Kết quả:

```
duc@duc:~/Assignment_2/source_code$ ./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  3
```


Time slot 14
Time slot 15
CPU 0: Put process 4 to run queue
CPU 0: Dispatched process 1
Time slot 16
Time slot 17
CPU 1: Processed 3 has finished
CPU 1: Dispatched process 2
Time slot 18
Time slot 19
CPU 0: Processed 1 has finished
CPU 0: Dispatched process 4
Time slot 20
Time slot 21
CPU 1: Processed 2 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: 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)
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)
014: 03800-03bff - PID: 03 (idx 000, nxt: 015)
015: 03c00-03fff - PID: 03 (idx 001, nxt: 016)
016: 04000-043ff - PID: 03 (idx 002, nxt: 017)
        041e7: 0a
017: 04400-047ff - PID: 03 (idx 003, nxt: 018)
018: 04800-04bff - PID: 03 (idx 004, nxt: -01)
019: 04c00-04fff - PID: 04 (idx 000, nxt: 020)
020: 05000-053ff - PID: 04 (idx 001, nxt: 021)
021: 05400-057ff - PID: 04 (idx 002, nxt: 022)
        055e7: 0a
022: 05800-05bff - PID: 04 (idx 003, nxt: 023)
023: 05c00-05fff - PID: 04 (idx 004, nxt: -01)
024: 06000-063ff - PID: 02 (idx 000, nxt: 025)
025: 06400-067ff - PID: 02 (idx 001, nxt: 026)
026: 06800-06bff - PID: 02 (idx 002, nxt: 027)
027: 06c00-06fff - PID: 02 (idx 003, nxt: -01)
032: 08000-083ff - PID: 03 (idx 000, nxt: 033)
033: 08400-087ff - PID: 03 (idx 001, nxt: 034)
034: 08800-08bff - PID: 03 (idx 002, nxt: 035)
035: 08c00-08fff - PID: 03 (idx 003, nxt: -01)
047: 0bc00-0bfff - PID: 01 (idx 000, nxt: -01)
```

- Giản đồ Gantt:



3.2 OS test 1

- Kết quả:

```
duc@duc:~/Assignment_2/source_code$ ./os os_1
Time slot   0
Time slot   1
        Loaded a process at input/proc/p0, PID: 1
        CPU 2: Dispatched process  1
Time slot   2
        Loaded a process at input/proc/s3, PID: 2
Time slot   3
        CPU 3: Dispatched process  2
        CPU 2: Put process  1 to run queue
        CPU 2: Dispatched process  1
Time slot   4
        Loaded a process at input/proc/m1, PID: 3
        CPU 0: Dispatched process  3
Time slot   5
        CPU 3: Put process  2 to run queue
        CPU 3: Dispatched process  2
        CPU 2: Put process  1 to run queue
        CPU 2: Dispatched process  1
Time slot   6
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  3
        Loaded a process at input/proc/s2, PID: 4
```

```
Time slot   7
        CPU 3: Put process  2 to run queue
        CPU 3: Dispatched process  2
        Loaded a process at input/proc/m0, PID: 5
        CPU 1: Dispatched process  4
        CPU 2: Put process  1 to run queue
        CPU 2: Dispatched process  5
Time slot   8
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  3
Time slot   9
        CPU 3: Put process  2 to run queue
        CPU 3: Dispatched process  1
        CPU 1: Put process  4 to run queue
        CPU 1: Dispatched process  4
        Loaded a process at input/proc/p1, PID: 6
        CPU 2: Put process  5 to run queue
        CPU 2: Dispatched process  2
Time slot  10
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  6
```

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

```
Time slot  16
        CPU 0: Processed  1 has finished
        CPU 0: Dispatched process  7
        Loaded a process at input/proc/s1, PID: 8
        CPU 2: Put process  6 to run queue
        CPU 2: Dispatched process  8
Time slot  17
        CPU 1: Put process  5 to run queue
        CPU 1: Dispatched process  5
        CPU 3: Put process  4 to run queue
        CPU 3: Dispatched process  6
Time slot  18
        CPU 1: Processed  5 has finished
        CPU 1: Dispatched process  4
        CPU 2: Put process  8 to run queue
        CPU 2: Dispatched process  8
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  7
Time slot  19
        CPU 3: Put process  6 to run queue
        CPU 3: Dispatched process  6
```

```
Time slot  20
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  7
        CPU 1: Put process  4 to run queue
        CPU 1: Dispatched process  4
        CPU 2: Put process  8 to run queue
        CPU 2: Dispatched process  8
Time slot  21
        CPU 3: Put process  6 to run queue
        CPU 3: Dispatched process  6
Time slot  22
        CPU 1: Processed  4 has finished
        CPU 1 stopped
        CPU 0: Put process  7 to run queue
        CPU 0: Dispatched process  7
        CPU 2: Put process  8 to run queue
        CPU 2: Dispatched process  8
Time slot  23
        CPU 3: Processed  6 has finished
        CPU 3 stopped
        CPU 2: Processed  8 has finished
        CPU 2 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: 06 (idx 000, nxt: 001)
001: 00400-007ff - PID: 06 (idx 001, nxt: 031)
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)
007: 01c00-01fff - PID: 05 (idx 000, nxt: 008)
        01fe8: 15
008: 02000-023ff - PID: 05 (idx 001, nxt: -01)
013: 03400-037ff - PID: 06 (idx 000, nxt: 014)
014: 03800-03bff - PID: 06 (idx 001, nxt: 015)
015: 03c00-03fff - PID: 06 (idx 002, nxt: 016)
016: 04000-043ff - PID: 06 (idx 003, nxt: -01)
021: 05400-057ff - PID: 01 (idx 000, nxt: -01)
024: 06000-063ff - PID: 05 (idx 000, nxt: 025)
        06014: 66
025: 06400-067ff - PID: 05 (idx 001, nxt: -01)
031: 07c00-07fff - PID: 06 (idx 002, nxt: 032)
        07de7: 0a
032: 08000-083ff - PID: 06 (idx 003, nxt: 033)
033: 08400-087ff - PID: 06 (idx 004, nxt: -01)
```

- Giản đồ Gantt: