

HO CHI MINH UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



OPERATING SYSTEM

Assignment

Simple Operating System

Lecturer: Tran Viet Toan

Students: Nguyen Trung Nguyen - 1852619

Thanh pho Ho Chi Minh, 5/2021

Contents

1	Scheduler	3
1.1	Question - Priority Feedback Queue	3
1.2	Result - Gantt Diagrams	3
1.3	Implementation	4
1.3.1	Priority Queue	4
1.3.2	Scheduler	4
2	Memory Management	5
2.1	Question - Segmentation with Paging	5
2.2	Result - Status of RAM	5
2.3	Implementation	7
2.3.1	Find paging table from segment	7
2.3.2	Mapping virtual addresses to physical addresses	7
2.3.3	Memory allocation	8
2.3.4	Memory recovery	8
3	Put it all together	9

List of Figures

1	Gantt diagram for test 0	3
2	Gantt diagram for test 1	4
3	Gantt diagram for Test 0	9
4	Gantt diagram for Test I	9

1 Scheduler

1.1 Question - Priority Feedback Queue

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

The Priority Feedback Queue (PFQ) algorithm uses the ideas of a number of other algorithms including Priority Scheduling - each process has a priority to execute, Multilevel Queue algorithm - uses multi-level process queues, Round Robin algorithm - uses quantum time for processes to execute.

Specifically, the PFQ algorithm uses two queues, *ready_queue* and *run_queue* with the following meanings:

- *ready_queue*: This queue contains processes which the CPU searches for. When it moves to the next slot.
- *run_queue*: This queue contains processes that are waiting to continue executing after running out of their slots without completing their process. Processes in this queue can only continue to the next slot when *ready_queue* is empty.
- Both queues are priority queues, the priority is based on the priority of the process in the queue.

preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

Advantages of using priority feedback queue

- With the idea of RR algorithm with a quantum time interval, creating fairness in execution time between processes, but it still executes in order of priority (using time for more important process first, avoid waiting time for users).
- Bring the idea of MLQS algorithm but use quantum time to solve the problem of starvation or indefinite blocking.
- It just has 2 main queues, so it is lower scheduling overhead than Multilevel Feedback Queue Scheduling although they use similar algorithm.
- Compare with First Come First Serve (FCFS) and Shortest Job Next (SJN), it is preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

1.2 Result - Gantt Diagrams

REQUIREMENT: Draw Gantt diagram describing how processes are executed by the CPU.

Test 0:

P1				P2		P1		P2		P1		P2		P1		P2		P1			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

<https://www.overleaf.com/project/60a335507b46d25d1d3c29a1>

Figure 1: Gantt diagram for test 0

Test 1:

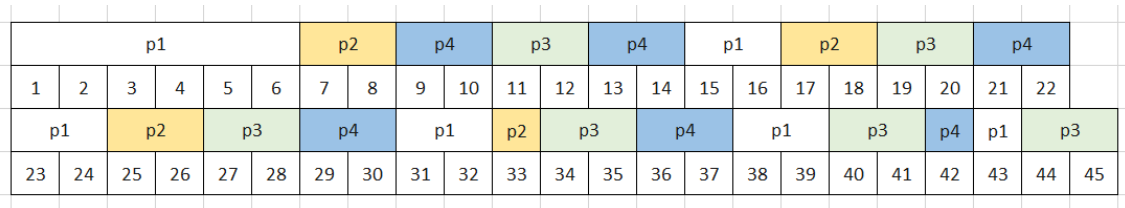


Figure 2: Gantt diagram for test 1

1.3 Implementation

1.3.1 Priority Queue

With the `enqueue()` function, the Priority Queue in this case handles no more than `MAX_QUEUE_SIZE` processes, so we simply need to use the loop to handle the functionality a priority queue needs.

With the `dequeue()` function, we find the process with the highest priority to return, and update the state of the queue when deleting an element.

1.3.2 Scheduler

With `get_proc` function, we need to check whether ready queue empty or not. If it is empty we need to move all process PCBs from run queue to ready queue. Then, we return the highest priority PCB in ready queue.

2 Memory Management

2.1 Question - Segmentation with Paging

QUESTION: What is the advantage and disadvantage of segmentation with paging?

Advantages of the algorithm

- Save memory, use memory efficiently.
- Have the advantages of paging algorithm:
 - Simplify memory allocation.
 - Fixed foreign fragmentation.
- Solve the external fragmentation problem of the segmentation algorithm by paging within each segment.

Disadvantages of Algorithm

- Internal fragmentation of the paging algorithm remains.

2.2 Result - Status of RAM

REQUIREMENT: Show the status of RAM after each memory allocation and deallocation function call.

In detail, the status of RAM in every steps:

Test 0:

Test 1:

2.3 Implementation

2.3.1 Find paging table from segment

In this assignment, each address is represented by 20 bits, where the first 5 bits are the segment, the next 5 bits are the page, and the last 10 bits are the offset.

This function takes in 5 bits segment *index* segment table *seg_table*, should find the paging table *res* of the corresponding segment in the aforementioned shard table.

Since the segment table *seg_table* is a list of *u* elements with the structure *(v_index, page_table_t)*, in that *v_index* is 5 bits segment of *u* elements and *page_table_t* is the corresponding page part table for that segment. So to find *res*, we just need to browse on this segment table, which *u* element has *v_index* equal to *index* to find, we return the corresponding *page_table*.

The implementation for the above function show below :

2.3.2 Mapping virtual addresses to physical addresses

Because each address consists of 20 bits with the above organization, so to create a physical address, we take the first 10 bits (segment and page) and connect the last 10 bits (offset). Each *page_table_t* stores elements where *p_index* is the first 10 bits. so to create a physical address, we just need to shift left those 10 bits by 10 bits offset and then or (—) these two strings.

The implementation for the above function show below :

2.3.3 Memory allocation

2.3.3.1 Check the readiness of memory: In this step we check if the memory is available both on the physical memory and on the logical memory.

On the physical area, we check the number of empty pages, not yet used by any process, if enough pages need to be allocated, the physical area is ready. In addition, to optimize the search time when falling into the case of insufficient memory, we can organize *mem_stat* as a list, including size management, free memory, ... for quick access to the required information.

On the logical memory area, we check based on the break point of the process, not exceeding the allowed memory area.

2.3.3.2 Allocate memory : Implementation steps:

- Browse the physical memory, find idle pages, assign this page to be used by the process.
- Create *last_allocated_page_index* variable to make updating *next* value easier.
- On the logical memory area, based on the allocated address, from the starting address and the page position, we find its segments and pages. From there update the corresponding paging and segment tables.

Implementation details are described below.

2.3.4 Memory recovery

2.3.4.1 Physical address revocation : Convert the logical address from the process to the physical, then based on the next value of the mem, we update the corresponding address string.

2.3.4.2 Logical address update : Based on the number of deleted pages on the block of the physical address, we find the pages on the logical address in turn, based on the address, we find the corresponding segment and page. Then update the paging table again, after the update process, if the table is empty, delete this table in the segment.

2.3.4.3 Update break point : Executed only when the last block on the logical address is deleted, then from there cycle back through the pages, until the page is being used, then stop.

3 Put it all together

Finally, we combine scheduler and Virtual Memory Engine to form a complete OS. Then we check work by first compiling the whole source code with following command "make all" and get results by running the following command "make test_all" which is saved in file "result/os".

The Gantt diagrams below represent case of "result/all" in source code.

Test 0:

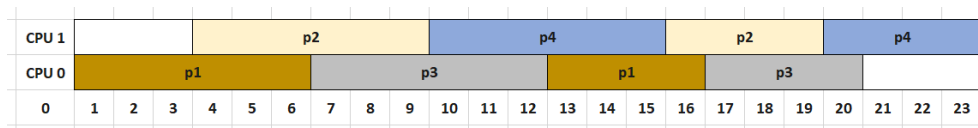


Figure 3: Gantt diagram for Test 0

Test 1:

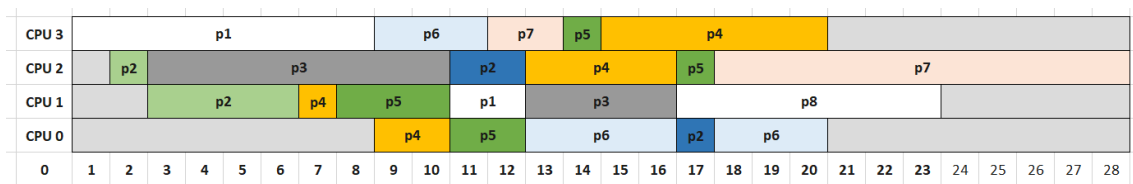


Figure 4: Gantt diagram for Test 1