Ho Chi Minh City University of Technology

FACULTY OF COMPUTER SCIENCE & ENGINEERING

Assignment

Operating System

Simple Operating System Report

Ho Chi Minh City, 6/2021

Group Members

Tên	MSSV	Proccess
Lê Tự Ngọc Minh	1952844	30%
Nguyễn Thành Lộc	1952330	30%
Võ Trần Minh Đạt	1913086	40%

1. Goals:

The objective of this assignment is the simulation of major components in a simple operating system, for example, scheduler, synchronization, related operations of physical memory and virtual memory .

2. Content:

In detail, student will practice with three major modules: scheduler, synchronization, mechanism of memory allocation from virtual-to-physical memory Oscilloscope (x1)

- Scheduler
- Synchronization
- the operations of mem-allocation from virtual-to-physical

3. Result:

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.

Contents

Short Introduction:

In computer science, a **multilevel feedback queue** is a scheduling algorithm. Solaris 2.6 Time-Sharing (TS) scheduler implements this algorithm.[1] The MacOS and Microsoft Windows schedulers can both be regarded as examples of the broader class of multilevel feedback queue schedulers.[2] This scheduling algorithm is intended to meet the following design requirements for multimode systems:

1. Give preference to short jobs.
2. Give preference to I/O bound processes.
3. Separate processes into categories based on their need for the processor.

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?

Priority Feedback Queue (PFQ) use the idea of some algorithms:

- Priority Scheduling : each process contains priority to execute
- Multilevel Queue : using queue to wait for next processes
- Round Robin: use quantum time for process to execute

Some Algorithms that we learn:

- First Come First Served (FCFS)
- Shortest Job First (SJF)
- Shortest Remaining Time First (SRTF)
- Priority Scheduling (PS)
- Round Robin (RR)
- Multilevel Queue Scheduling (MLQS)
- Multilevel Feedback Queue (MLFQ)

In Specific, PFQ use 2 queues:

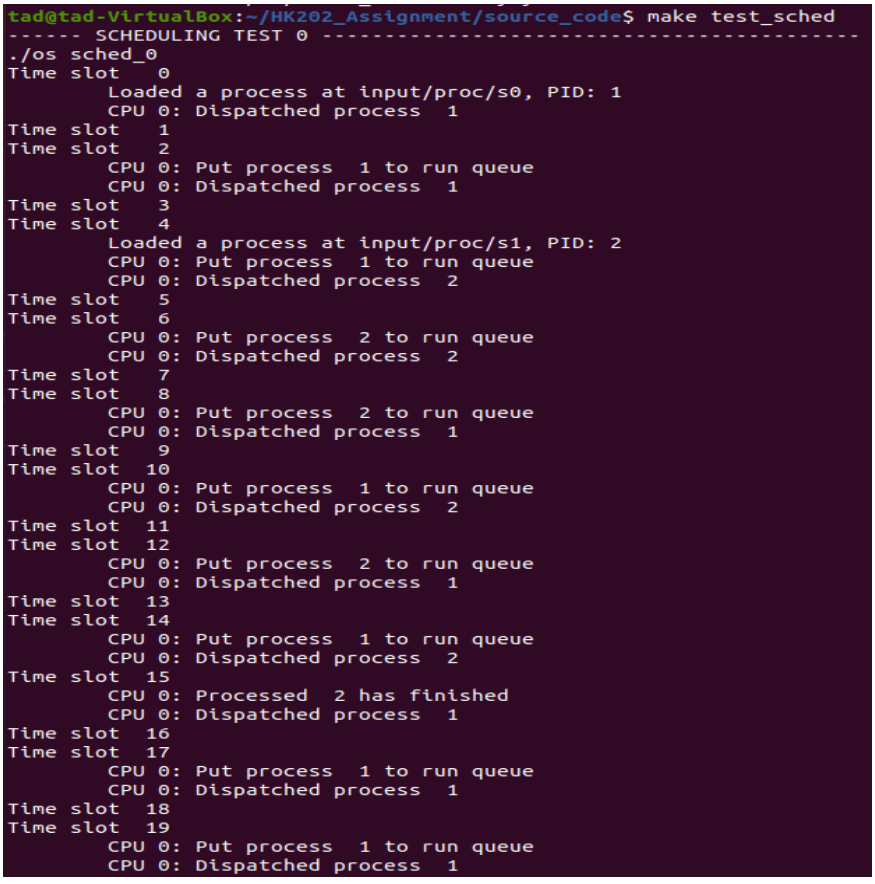
- ready_queue: hold processes that have higher priority level than run_queue. When CPU changes state, it will get processes in this queue.
- run_queue: hold processes which waiting to continue to execute after doing without complement. Processes in this queue waiting to put into ready_queue.
- Both of them are priority queues, priority level depends on each process.

Advantages of PFQ Algorithm

- Using timeslot, which is the idea of RR algorithm in the interval of quantum time. It makes equality in execution time between processes, avoid CPU usage, indefinitely delay.
- Using queues, which is the idea of MLQS and MLFQ algorithms. Two queues are moved mutually between processes until finish. Increasing execution time for each process.
- The equality between processes is guaranteed, depend on their priority level of them.

1.2 Result – Gantt Diagram

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

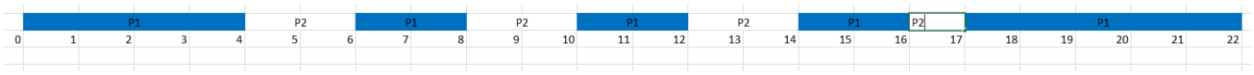


```
Time slot  19
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot  20
Time slot  21
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot  22
        CPU 0: Processed  1 has finished
        CPU 0 stopped

MEMORY CONTENT:
NOTE: Read file output/sched_0 to verify your result
------ SCHEDULING TEST 1 ---------------------------------------------
./os sched_1
Time slot   0
        Loaded a process at input/proc/s0, PID: 1
        CPU 0: Dispatched process  1
Time slot   1
Time slot   2
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  1
Time slot   3
Time slot   4
        Loaded a process at input/proc/s1, PID: 2
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  2
Time slot   5
Time slot   6
        Loaded a process at input/proc/s2, PID: 3
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  3
Time slot   7
        Loaded a process at input/proc/s3, PID: 4
Time slot   8
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  4
Time slot   9
Time slot  10
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  2
Time slot  11
Time slot  12
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  3
Time slot  13
Time slot  14
```

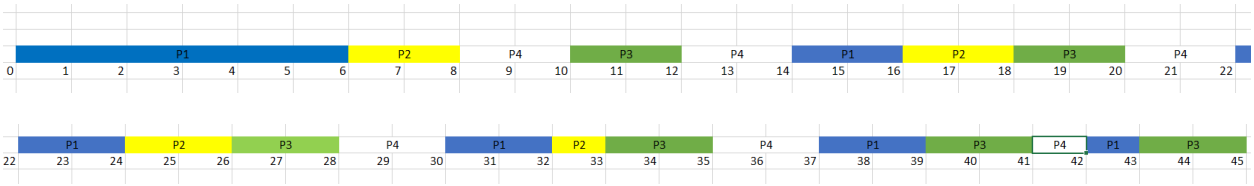
```
Time slot   14
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  1
Time slot   15
Time slot   16
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot   17
Time slot   18
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  2
Time slot   19
Time slot   20
        CPU 0: Put process  2 to run queue
        CPU 0: Dispatched process  3
Time slot   21
Time slot   22
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  1
Time slot   23
Time slot   24
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot   25
Time slot   26
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  2
Time slot   27
        CPU 0: Processed  2 has finished
        CPU 0: Dispatched process  3
Time slot   28
Time slot   29
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  1
Time slot   30
Time slot   31
        CPU 0: Put process  1 to run queue
        CPU 0: Dispatched process  4
Time slot   32
Time slot   33
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  3
Time slot   34
Time slot   35
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  1
```

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

MEMORY CONTENT:
NOTE: Read file output/sched_1 to verify your result
```

Gantt Diagram:/sched0



Gantt Diagram:/sched1



2. Memory Management

2.1. Question - Segmentation with Paging:

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

Advantages	***Disadvantages***
No external fragmentation	Internal fragmentation remains a problem.
Reduced memory requirements as no. of pages limited to segment size.	Hardware is more complex than segmented paging.
Page table size is smaller just like segmented paging	Extra level of paging at first stage adds to the delay in memory access.
Similar to segmented paging, the entire segment need not be swapped out.	

2.2 Result of status of Ram:

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

This is the result of process after allocation and deallocation instruction:

Test 0:

```
1 ============ Allocation ============
2 000: 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
3 001 : 00400-007ff - PID : 01 ( idx 001 , nxt : 002)
4 002 : 00800-00 bff - PID : 01 ( idx 002 , nxt : 003)
```

```
003 : 00 c00-00fff - 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 : 01800-01 bff - PID : 01 ( idx 006 , nxt : 007)
007 : 01 c00-01fff - 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 : 02800-02 bff - PID : 01 ( idx 010 , nxt : 011)
011 : 02 c00-02fff - 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 ============
000 : 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
001 : 00400-007ff - PID : 01 ( idx 001 , nxt : 002)
002 : 00800-00 bff - PID : 01 ( idx 002 , nxt : 003)
003 : 00 c00-00fff - 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 : 01800-01 bff - PID : 01 ( idx 006 , nxt : 007)
007 : 01 c00-01fff - 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 : 02800-02 bff - PID : 01 ( idx 010 , nxt : 011)
011 : 02 c00-02fff - 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 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
============ Deallocation ============
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
============ Allocation ============
000 : 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
001 : 00400-007ff - PID : 01 ( idx 001 , nxt : -01)
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
============ Allocation ============
000 : 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
001 : 00400-007ff - PID : 01 ( idx 001 , nxt : -01)
002 : 00800-00 bff - PID : 01 ( idx 000 , nxt : 003)
003 : 00 c00-00fff - 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 : 01800-01 bff - PID : 01 ( idx 004 , nxt : -01)
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)

============ Final - dump ( ) ============
000 : 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
003 e8 : 15
001 : 00400-007ff - PID : 01 ( idx 001 , nxt : -01)
002 : 00800-00 bff - PID : 01 ( idx 000 , nxt : 003)
003 : 00 c00-00fff - 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 : 01800-01 bff - PID : 01 ( idx 004 , nxt : -01)
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
03814: 66
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
```

Test 1:

```
========== Allocation ==========
000: 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
001 : 00400-007ff - PID : 01 ( idx 001 , nxt : 002)
002 : 00800-00 bff - PID : 01 ( idx 002 , nxt : 003)
003 : 00 c00-00fff - 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 : 01800-01 bff - PID : 01 ( idx 006 , nxt : 007)
007 : 01 c00-01fff - 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 : 02800-02 bff - PID : 01 ( idx 010 , nxt : 011)
011 : 02 c00-02fff - 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 ==========
000 : 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
001 : 00400-007ff - PID : 01 ( idx 001 , nxt : 002)
002 : 00800-00 bff - PID : 01 ( idx 002 , nxt : 003)
003 : 00 c00-00fff - 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 : 01800-01 bff - PID : 01 ( idx 006 , nxt : 007)
007 : 01 c00-01fff - 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 : 02800-02 bff - PID : 01 ( idx 010 , nxt : 011)
011 : 02 c00-02fff - 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 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
========== Deallocation ==========
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
========== Allocation ==========
000 : 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
001 : 00400-007ff - PID : 01 ( idx 001 , nxt : -01)
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
========== Allocation ==========
000 : 00000-003ff - PID : 01 ( idx 000 , nxt : 001)
001 : 00400-007ff - PID : 01 ( idx 001 , nxt : -01)
002 : 00800-00 bff - PID : 01 ( idx 000 , nxt : 003)
003 : 00 c00-00fff - 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 : 01800-01 bff - PID : 01 ( idx 004 , nxt : -01)
```

```
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
============ Deallocation ============
002 : 00800-00 bff - PID : 01 ( idx 000 , nxt : 003)
003 : 00 c00-00fff - 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 : 01800-01 bff - PID : 01 ( idx 004 , nxt : -01)
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
============ Deallocation ============
014 : 03800-03 bff - PID : 01 ( idx 000 , nxt : 015)
015 : 03 c00-03fff - PID : 01 ( idx 001 , nxt : -01)
============ Deallocation ============

============ Final - dump ( ) ============
```

3. Put it all together

3.1 Interpret the operation

Multiple FIFO queues are used and the operation is as follows:

1. A new process is inserted at the end (tail) of the top-level FIFO queue.
2. At some stage the process reaches the head of the queue and is assigned the CPU.
3. If the process is completed within the time quantum of the given queue, it leaves the system.
4. If the process voluntarily relinquishes control of the CPU, it leaves the queuing network, and when the process becomes ready again it is inserted at the tail of the same queue which it relinquished earlier.
5. If the process uses all the quantum time, it is pre-empted and inserted at the end of the next lower level queue. This next lower level queue will have a time quantum which is more than that of the previous higher level queue.
6. This scheme will continue until the process completes or it reaches the base level queue.

    - At the base level queue the processes circulate in round robin fashion until they complete and leave the system. Processes in the base level queue can also be scheduled on a first come first served basis.
    - Optionally, if a process blocks for I/O, it is 'promoted' one level, and placed at the end of the next-higher queue. This allows I/O bound processes to be favored by the scheduler and allows processes to 'escape' the base level queue.

For scheduling, the scheduler always starts picking up processes from the head of the highest level queue. Only if the highest level queue has become empty will the scheduler take up a process from the next lower level queue. The same policy is implemented for picking up in

the subsequent lower level queues. Meanwhile, if a process comes into any of the higher level queues, it will preempt a process in the lower level queue.

Also, a new process is always inserted at the tail of the top level queue with the assumption that it will complete in a short amount of time. Long processes will automatically sink to lower level queues based on their time consumption and interactivity level. In the multilevel feedback queue a process is given just one chance to complete at a given queue level before it is forced down to a lower level queue.

3.2 Result

After combining sched.c and mem.c, we use make all and have result:

```
----- 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
```

```
----- OS TEST 1 -------------------------------------------------
./os os_1
Time slot   0
Time slot   1
        Loaded a process at input/proc/p0, PID: 1
        CPU 1: Dispatched process  1
Time slot   2
        Loaded a process at input/proc/s3, PID: 2
Time slot   3
        CPU 2: Dispatched process  2
        CPU 1: Put process  1 to run queue
        CPU 1: Dispatched process  1
Time slot   4
        Loaded a process at input/proc/m1, PID: 3
        CPU 0: Dispatched process  3
Time slot   5
        CPU 2: Put process  2 to run queue
        CPU 2: Dispatched process  2
        CPU 1: Put process  1 to run queue
        CPU 1: Dispatched process  1
Time slot   6
        Loaded a process at input/proc/s2, PID: 4
        CPU 0: Put process  3 to run queue
        CPU 0: Dispatched process  4
Time slot   7
        CPU 3: Dispatched process  3
        CPU 2: Put process  2 to run queue
        CPU 1: Put process  1 to run queue
        CPU 1: Dispatched process  1
        Loaded a process at input/proc/m0, PID: 5
        CPU 2: Dispatched process  2
Time slot   8
        CPU 0: Put process  4 to run queue
        CPU 0: Dispatched process  5
Time slot   9
        CPU 3: Put process  3 to run queue
        CPU 3: Dispatched process  4
        CPU 2: Put process  2 to run queue
        CPU 2: Dispatched process  3
        CPU 1: Put process  1 to run queue
        CPU 1: Dispatched process  2
        Loaded a process at input/proc/p1, PID: 6
Time slot  10
        CPU 0: Put process  5 to run queue
        CPU 0: Dispatched process  1
Time slot  11
        CPU 3: Put process  4 to run queue
```

```
MEMORY CONTENT:
000: 00000-003ff - PID: 05 (idx 000, nxt: 001)
001: 00400-007ff - PID: 05 (idx 001, nxt: 009)
002: 00800-00bff - PID: 06 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 06 (idx 001, nxt: 004)
004: 01000-013ff - PID: 06 (idx 002, nxt: 005)
        011e7: 0a
005: 01400-017ff - PID: 06 (idx 003, nxt: 006)
006: 01800-01bff - PID: 06 (idx 004, nxt: -01)
007: 01c00-01fff - PID: 05 (idx 000, nxt: 008)
        01fe8: 15
008: 02000-023ff - PID: 05 (idx 001, nxt: -01)
009: 02400-027ff - PID: 05 (idx 002, nxt: 010)
010: 02800-02bff - PID: 05 (idx 003, nxt: 011)
011: 02c00-02fff - PID: 05 (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
029: 07400-077ff - PID: 05 (idx 000, nxt: 030)
        07414: 66
030: 07800-07bff - PID: 05 (idx 001, nxt: -01)
NOTE: Read file output/os_1 to verify your result
```

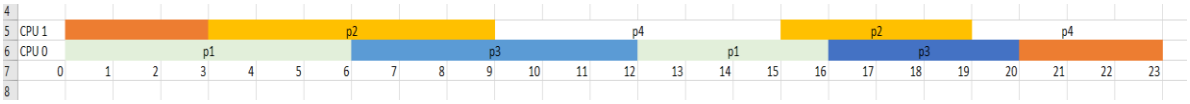
```
Time slot  22
Time slot  23
        CPU 0: Processed  4 has finished
        CPU 0 stopped

MEMORY CONTENT:
000: 00000-003ff - PID: 01 (idx 000, nxt: -01)
        00811: 24
0       - PID: 03 (idx 000, nxt: 002)
002: 00800-00bff - PID: 03 (idx 001, nxt: 003)
003: 00c00-00fff - PID: 03 (idx 002, nxt: 004)
004: 01000-013ff - PID: 03 (idx 003, nxt: -01)
005: 01400-017ff - PID: 04 (idx 000, nxt: 006)
006: 01800-01bff - PID: 04 (idx 001, 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: 04 (idx 002, nxt: 013)
013: 03400-037ff - PID: 04 (idx 003, nxt: -01)
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)
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)
057: 0e400-0e7ff - PID: 04 (idx 000, nxt: 058)
058: 0e800-0ebff - PID: 04 (idx 001, nxt: 059)
059: 0ec00-0efff - PID: 04 (idx 002, nxt: 060)
        0ede7: 0a
060: 0f000-0f3ff - PID: 04 (idx 003, nxt: 061)
061: 0f400-0f7ff - PID: 04 (idx 004, nxt: -01)
NOTE: Read file output/os_0 to verify your result
```

```
Time slot  19
        CPU 2: Put process  7 to run queue
        CPU 2: Dispatched process  8
        CPU 3: Processed  5 has finished
        CPU 3: Dispatched process  7
        CPU 1: Put process  6 to run queue
        CPU 1: Dispatched process  6
Time slot  20
        CPU 0: Processed  4 has finished
        CPU 0 stopped
Time slot  21
        CPU 2: Put process  8 to run queue
        CPU 2: Dispatched process  8
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  7
        CPU 1: Put process  6 to run queue
        CPU 1: Dispatched process  6
Time slot  22
Time slot  23
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  7
        CPU 2: Put process  8 to run queue
        CPU 2: Dispatched process  8
        CPU 1: Processed  6 has finished
        CPU 1 stopped
Time slot  24
        CPU 2: Processed  8 has finished
        CPU 2 stopped
Time slot  25
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  7
Time slot  26
Time slot  27
        CPU 3: Put process  7 to run queue
        CPU 3: Dispatched process  7
Time slot  28
        CPU 3: Processed  7 has finished
        CPU 3 stopped

MEMORY CONTENT:
000: 00000-003ff - PID: 05 (idx 000, nxt: 001)
001: 00400-007ff - PID: 05 (idx 001, nxt: 009)
002: 00800-00bff - PID: 06 (idx 000, nxt: 003)
003: 00c00-00fff - PID: 06 (idx 001, nxt: 004)
004: 01000-013ff - PID: 06 (idx 002, nxt: 005)
        011e7: 0a
005: 01400-017ff - PID: 06 (idx 003, nxt: 006)
006: 01800-01bff - PID: 06 (idx 004, nxt: -01)
```

```
Time slot  11
        CPU 3: Put process  4 to run queue
        CPU 3: Dispatched process  6
        CPU 2: Put process  3 to run queue
        CPU 2: Dispatched process  4
        Loaded a process at input/proc/s0, PID: 7
        CPU 1: Put process  2 to run queue
        CPU 1: Dispatched process  7
Time slot  12
        CPU 0: Processed  1 has finished
        CPU 0: Dispatched process  5
Time slot  13
        CPU 2: Put process  4 to run queue
        CPU 2: Dispatched process  3
        CPU 3: Put process  6 to run queue
        CPU 3: Dispatched process  4
        CPU 1: Put process  7 to run queue
        CPU 1: Dispatched process  2
Time slot  14
        CPU 0: Put process  5 to run queue
        CPU 0: Dispatched process  6
Time slot  15
        CPU 2: Processed  3 has finished
        CPU 2: Dispatched process  7
        CPU 3: Put process  4 to run queue
        CPU 3: Dispatched process  5
        CPU 1: Put process  2 to run queue
        CPU 1: Dispatched process  4
Time slot  16
        Loaded a process at input/proc/s1, PID: 8
        CPU 0: Put process  6 to run queue
        CPU 0: Dispatched process  8
Time slot  17
        CPU 3: Put process  5 to run queue
        CPU 3: Dispatched process  2
        CPU 2: Put process  7 to run queue
        CPU 2: Dispatched process  7
        CPU 1: Put process  4 to run queue
        CPU 1: Dispatched process  6
Time slot  18
        CPU 3: Processed  2 has finished
        CPU 3: Dispatched process  5
        CPU 0: Put process  8 to run queue
        CPU 0: Dispatched process  4
Time slot  19
        CPU 2: Put process  7 to run queue
        CPU 2: Dispatched process  8
```

Here are Gantt diagrams of all testcases in src file:

Text 0:



Test1: