**Algorithm test 2021.11.17**

**Question 1:**

Given a real-time OS scheduling, process P0, P1, P2, P3 have time periods 3, 5, 6, 10 respectively, run on 2 CPU cores, refer to the PDF document

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Core 1 | P0 | P2 | P2 | P3 | P0 | P3 | P0 | | P0 | P3 | P0 | P3 | P0 | P3 |
| Core 2 | P1 | | P0 | P1 | | P2 | | P1 | | P2 | | P1 | | P2 |

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

I forgot exactly the diagram, somehow both P0 and P2 should appear somewhere in Core 2

Give short answers with brief justification

1. Determine whether it is partitioned scheduling or global scheduling
2. Determine whether it is RM, DM or EDF
3. Given that the deadline aligns with the incoming time period, is this schedulable?

**Answer:**

1. Global scheduling, P0 and P2 runs on both cores
2. EDF, if it is fixed priority, then since P0 and P1 runs at t=0, they have higher priority than P2 and P3, however at **t=9 (very important!!)**, P0 does not start running but given priority to P3, so it cannot be fixed priority
3. Schedulable, didn’t miss deadline

**Question 2:**

Resource diagram, if P4 requests (A=1, B=1, C=1), then what is the minimum x and y to make this become a safe state? Explain with explanation on a safe sequence algorithm.

|  |  |  |
| --- | --- | --- |
|  |  | Availability:  A B C  x 1 y |
| Process | Allocation | Need |
| P0 | Forgot | Forgot, but certain that B>1 |
| P1 | Forgot | Forgot, but certain that B=1 |
| P2 | Forgot | 2 0 1 |
| P3 | Forgot | Forgot, but certain that B>1 |
| P4 | 1 1 1 | 4 1 3 |

**Answer:** x = 3 ; y = 2

After the request is serviced, then P4 becomes: Alocation: 2 2 2 Need: 3 0 2

Available resources become: x-1 0 y-1

Select P2 first, x-1>=2 and y-1>=1, so x = 3 and y = 2 as minimum values of x and y.

**Question 3:**

Given the following, is Progress and Bounded Waiting satisfied? Give a brief justification

|  |  |
| --- | --- |
| P0 | P1 |
| While (1) {  turn = 1;  flag = False;  while (turn==1 && flag==True);  **critical section;**  flag = True;  **remaining section;**  } | While (1) {  turn = 0;  flag = True;  while (turn==0 && flag==False);  **critical section;**  flag = False;  **remaining section;**  } |

**Answer:**

Progress and Bounded Waiting are both satisfied.

Bounded Waiting: If need to block P0, then turn==1 && flag==True, but P1 completes will flip turn to False, so P1 will not block P0 from entering the critical section. Same vice versa.

Progress: If block P0, then it will be the case that turn==1 && flag==True, but if cannot be the case permanently, e.g. P1 will flip flag to False after leaving the critical section or P0 will turn flag to False before the while loop. So P0 is not blocked permanently. Same for P1.

**Question 4:**

2 level page tables, the outer page table has 8 entries, each entry takes 4 bytes, the outer page table fits exactly into one page.

1. What is the size of each page?
2. How many logical addresses?
3. What is the format of a logical address?

**Answer:** **I also don’t know, I put in a wrong answer in the exam**

1. 8 X 4 = 32 entries, so page size is 2^5
2. Outer page table has 2^3 entries, inner page table has 2^5 entries, page size = offset is 2^5 bytes, so total is 2^(3 + 5 + 5) = 2^13 number of logical addresses
3. Each logical address is a 13-bits string, first 3 bits denote a look-up in the outer page table, next 5 bits denote a look-up in the inner page table, remaining 5 bits denote offset

**Question 5:**

4 pages and 3 frames, the first 3 page loadings are given as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 2 | 3 |
| Frame 1 | 1 | 1 | 1 |
| Frame 2 |  | 2 | 2 |
| Frame 3 |  |  | 3 |

For the remaining page loading, given that the page out are: N N 3 1

(N means no page fault)

Ask: What is the page loading string if the following algorithms are implemented?

1. Clock
2. LRU

**Answer:**

1. 1 2 4 3 or 2 1 4 3
2. 1 2 4 3

**Question 6:**

Block size is 512 bytes and each pointer is 4 byte. If (i) contiguous (ii) linked (iii) indexed algorithm is implemented to retrieve a data of 200 bytes from a file of 2,000 bytes, how many I/O operations are needed?

**Answer:** **I also don’t know, I put in a wrong answer in the exam**

1. Contiguous: 2 operations, since need to access 2 blocks?
2. Linked: Ceiling(2200 / 508) = 5?
3. Indexed: 3 operations, since need to access 2 blocks + initial index table lookup?