**Question 1 (30 Points).**

Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds) and priority (0 is the highest priority) shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival time** | **Burst time** | **Priority** |
| P1 | 0 | 11 | 2 |
| P2 | 5 | 28 | 0 |
| P3 | 12 | 2 | 3 |
| P4 | 2 | 10 | 1 |
| P5 | 9 | 16 | 4 |

Draw the Gantt chart that illustrates the execution of these processes using **preemptive Priority scheduling** algorithm. Do the calculation of average turnaround time, and average waiting time. **(20+10 points)**

0 2 5 33 40 49 51 67

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P1 | P4 | P2 | P4 | P1 | P3 | P5 |

**- Average turnaround time:**

Pt(i) = end – arrival (pt(i): pi of turnaround)

P1: 49 – 0 = 49

P2: 33 – 5 = 28

P3: 51 – 12 = 39

P4: 40 – 2 = 38

P5: 67 – 9 = 58

=> average turnaround time =

**- Average waiting time:**

Pw(i) = pt(i) – burst (pw(i): pi of waiting)

P1: 49 – 11 = 38

P2: 28 – 28 = 0

P3: 39 – 2 = 37

P4: 38 – 10 = 28

P5: 58 – 16 = 42

=> average waiting time =

**Question 2 (30 Points). – 4 lab 5**

A. An operating system uses the banker's algorithm for deadlock avoidance when managing the allocation of five resource types A, B, C and D to five processes P0, P1, P2, P3 and P4. The table given below presents the current system state. Here, the Allocation matrix shows the current number

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Allocation** | | | | **Max** | | | | **Available** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 4 | 0 | 0 | 1 | 6 | 0 | 1 | 2 | 3 | 2 | 1 | 1 |
| **P1** | 1 | 1 | 0 | 0 | 1 | 7 | 5 | 0 |  |  |  |  |
| **P2** | 1 | 2 | 5 | 4 | 2 | 3 | 6 | 6 |  |  |  |  |
| **P3** | 0 | 5 | 3 | 3 | 1 | 6 | 5 | 3 |  |  |  |  |
| **P4** | 0 | 2 | 1 | 2 | 1 | 6 | 5 | 6 |  |  |  |  |

of resources of each type allocated to each process and the Max matrix shows the maximum number of resources of each type required by each process during its execution.

Answer the following questions using the Banker's algorithm:

(i) What is the content of the matrix Need?

(ii) Find if the system is in safe state. If it is, find the safe sequence. Show the detailed steps. **(3+22 Points)**

**=> Show bằng code ở dưới là lời giải tay**

(i) matrix Need

* Need matrix:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Allocation** | | | | **Max** | | | | **Available** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 4 | 0 | 0 | 1 | 6 | 0 | 1 | 2 | ~~3~~ | ~~2~~ | ~~1~~ | ~~1~~ | 2 | 0 | 1 | 1 |
| **P1** | 1 | 1 | 0 | 0 | 1 | 7 | 5 | 0 | ~~7~~ | ~~2~~ | ~~1~~ | ~~2~~ | 0 | 6 | 5 | 0 |
| **P2** | 1 | 2 | 5 | 4 | 2 | 3 | 6 | 6 | ~~8~~ | ~~4~~ | ~~6~~ | ~~6~~ | 1 | 1 | 1 | 2 |
| **P3** | 0 | 5 | 3 | 3 | 1 | 6 | 5 | 3 | ~~8~~ | ~~9~~ | ~~9~~ | ~~9~~ | 1 | 1 | 2 | 0 |
| **P4** | 0 | 2 | 1 | 2 | 1 | 6 | 5 | 6 | ~~8~~ | ~~11~~ | ~~10~~ | ~~11~~ | 1 | 4 | 4 | 4 |
|  |  |  |  |  |  |  |  |  | 9 | 12 | 10 | 11 |  |  |  |  |

(ii) safe state:

* Need(i)<=available -> new available = available + allocation(i)

P0: 2 0 1 1 <= 3 2 1 1 -> true -> available = 3 2 1 1 + 4 0 0 1 = 7 2 1 2

P1: 0 6 5 0 <= 7 2 1 2 -> false

P2: 1 1 1 2 <= 7 2 1 2 -> true -> available = 7 2 1 2 + 1 2 5 4 = 8 4 6 6

P3: 1 1 2 0 <= 8 4 6 6 -> true -> available = 8 4 6 6 + 0 5 3 3 = 8 9 9 9

P4: 1 4 4 4 < = 8 9 9 9 -> true -> available = 8 9 9 9 + 0 2 1 2 = 8 11 10 11

P1: 0 6 5 0 <= 8 11 10 11 -> true -> availabale = 8 11 10 11 + 1 1 0 0 = 9 12 10 11

=> [ P0, P2, P3, P4, P1 ] : it is safe

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Process P1 | Process P2 | | P(T);  Print ‘0’;  Print ‘1’;  V(T); | P(S);  Print ‘1’;  Print ‘0’;  V(S); |   B. Consider the two processes P1 and P2. Let us assume S and T are binary semaphores. S is initially 0 and Tis initially 1. In the code, P(S) and P(T) means wait operation; and V(S) and V(T) means signal operation. What will be the output string? Write the output only. Do not need to give explanation. **(5 Points)** |

=> Output: 01

|  |
| --- |
| Explan:  - s=0 -> p(s) not run  - T=1 -> p(t) -> t-- -> t =0  Print(“01”)  V(t) -> t++ -> t=1 |

**Question 3 (25 Points).**

A. Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3. Assuming there are 4 frames. Find numbers of page faults for the "Optimal page replacement algorithm". **(20 Points)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 3 |
| 7 | 7 | 7 | 7 | 7 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  | 1 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
|  |  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| yes | yes | yes | yes | no | **yes** | no | **yes** | no | no | no | no | no | no |

=> numbers of page faults : 6

|  |
| --- |
| Giải thích:  7 ko tồn tại trong dãy {3, 0, 4, 2, 3, 0, 3, 2, 3} nên thay thế 7  1 ko tồn tại trong dãy {4, 2, 3, 0, 3, 2, 3} nên thay thế 1  Nếu các số điều xuất hiện thì thay thế số xuất hiện xa nhất (chỉ tính lần xuất hiện đầu tiên)  Vd: {2, 1, 3, 2, 2, 1}  Thì thay thế số 3 |

B. Using a page size of 1024 bytes, find the physical address if the logical address is 2456.

Consider the following page table. **(5 Points)**

|  |  |
| --- | --- |
| **Page** | **Frame** |
| 0 | 3 |
| 1 | 6 |
| 2 | 5 |
| 3 | 1 |

|  |
| --- |
| Page Number = logical address / page size (chia lấy nguyên)  Offset = logical address % page size (chia lấy dư)  Physical Address = (frame number \* page size) + offset |

Page Number = 2456 / 1024 = 2

Offset = 2456 % 1024 = 408

Physical Address = (5 \* 1024) + 408 = 5528

**Question 4 (15 Points).**

Suppose that a disk drive has 200 cylinders, numbered 0 to 199. Disk requests come into the disk driver for cylinders: 82, 170, 43, 140, 24, 16, and 190 in that order. The disk head is currently positioned over the cylinder 50. A seek takes 3 millisecond per cylinder move. What is the total seek time using the algorithm SSTF?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 16 | 24 | 43 | 50 | 82 | 140 | 170 | 190 | 199 |
|  |  |  |  | (0) |  |  |  |  |  |
|  |  |  | (0) |  |  |  |  |  |  |
|  |  | (0) |  |  |  |  |  |  |  |
|  | (0) |  |  |  |  |  |  |  |  |
|  |  |  |  |  | (0) |  |  |  |  |
|  |  |  |  |  |  | (0) |  |  |  |
|  |  |  |  |  |  |  | (0) |  |  |
|  |  |  |  |  |  |  |  | (0) |  |

|  |  |
| --- | --- |
| **Move** | **Distance** |
| 50 -> 43 | 50 – 43 = 7 |
| 43 -> 24 | 43 – 24 = 19 |
| 24 -> 16 | 24 – 16 = 8 |
| 16 -> 82 | 82 – 16 = 66 |
| 82 -> 140 | 140 – 82 = 58 |
| 140 -> 170 | 170 – 140 = 30 |
| 170 -> 190 | 190 – 170 = 20 |

=> Total seek time = (7 + 19 + 8 + 66 + 58 + 30 + 20) \* 3 = 208 \* 3 = 624 ms