Assignment 9

Operating System Lab (CS341)

Department of CSE

IIT Patna

Date:- 07-April-2020

Instructions:

1. All the assignments should be completed and uploaded by 5.30 pm **(8th April)**. Marks will be deducted for the submissions made after 5.30 pm.
2. Markings will be based on the correctness and soundness of the outputs. Marks will be deducted in case of plagiarism.
3. Proper indentation and appropriate comments (if necessary) are mandatory. [2+2 marks]
4. You should zip all the required files and name the zip file as roll_no.zip, eg. 1501cs11.zip.
5. **Code for each question should be named as Q1_1.c, Q1_2.c, Q2.c etc.**
6. Each question will be evaluated on 10 test cases and marks will be given accordingly.
7. Upload your assignment (the zip file) in the following link:
https://www.dropbox.com/request/hxGMiCtKPC5ZGGmOIB3J

1. Write a C/C++ program to simulate the (i) MFT (Multiprogramming with a Fixed number of Tasks) and (ii) MVT (Multiprogramming with a Variable number of Tasks) memory management techniques.

Please follow the input and output format. Each integer value in the output has a meaning.
For (i) Consider all the input as command line arguments in the same sequence mentioned in Input MFT.

Input MFT:

Enter the total memory available (in Bytes) -- 1000
Enter the block size (in Bytes)-- 300
Enter the number of processes – 5 (P1-P5)
Enter memory required for process 1 (in Bytes) -- 275
Enter memory required for process 2 (in Bytes) -- 400
Enter memory required for process 3 (in Bytes) -- 290
Enter memory required for process 4 (in Bytes) -- 293
Enter memory required for process 5 (in Bytes) -- 100

No. of Blocks available in memory -- 3

Output Format:

P1 P2 P3 P4	(Process Ids)
1 0 1 1	(1= Memory Allocated, 0=Memory not allocated to that process)
25 NA 10 7	(Internal Fragmentation)
1	(Number of Processes that cannot be accommodated, 0 indicates all Processes have been accommodated)
42	(Total Internal Fragmentation)
100	(Total External Fragmentation)

INPUT MVT:

Enter the total memory available (in Bytes) -- 1000
Enter memory required for process 1 (in Bytes) -- 400
Memory is allocated for Process 1
Do you want to continue(y/n) -- y
Enter memory required for process 2 (in Bytes) -- 275
Memory is allocated for Process 2
Do you want to continue(y/n) -- y
Enter memory required for process 3 (in Bytes) -- 550

Output:

1	(Print 1, only If Insufficient memory)
P1 P2	(Process Ids)
400 275	(Allocated Memory)
325	(Total External Fragmentation)

Q:2 Assume that the main memory (physical memory) size is 2^32 bits (address range 0 to 2^32-1) and frame size is 2^17 bits. Design a MMU. It should have following methods:

1. add_process(process_id, req_memory) (time complexity <= O(n))
   : process_id (int) -> process id
   : req_memory (int) -> required memory in bits
   1.1 Use paging to allocate memory
   1.2 For each page allocate the first available frame
2. get_address(virtual_address) (time complexity must be O(1))
   : virtual_address (32 bit int): [5-bits:process_id,10-bits:page_offset,17-bits:frame_offset]
   2.1 Return the bit address for main memory

Use the above designed system to answer the queries for below inputs.

Input/Output:

First line contains an integer K.

Next line has 2*K integers specifying K frame ranges (inclusive) which cannot be allocated to any process.

Next line contains integer Q specifying the number of queries.

Next Q lines contain queries.

There are two types of queries:

1. a process_id req_memory
   add a process with process id 'process_id' and physical memory of 'req_memory' bits.
2. d virtual_address
   given virtual address 'virtual_address' print corresponding physical memory address.

You can assume that all the inputs provided will be in correct format and ranges. For each query of type ‘a’ process id will be different. For each query of type ‘d’ virtual address will be valid.

Sample:

Input:

```
3
0 10 15 20 30 40
3
a 1 64
d 134217735
d 134217748
```

Output:

```
24
48
```

Explanation:

Frame ranges (0, 10), (15, 20), (30, 40) cannot be allocated. Frame ranges are inclusive.

a 1 64

add process with id ‘1’ and physical memory of 64 bits.

d 134217735 (`00001 0000000000 00000000000000111`) (1, 0, 7) -> address of 7th bit of frame corresponding to 0th page for process with id ‘1’

d 134217748 (`00001 0000000000 00000000000010100`) (1, 0, 20) -> address of 20th bit of frame corresponding to 0th page for process with id ‘1’