Memory Manager Test Interface Requirements

CSE325

For your memory manager, I am providing the specifications of a test interface. Your project should be able to accept commands, as defined in this document, as a way to exercise your memory manager.

The commands listed are not necessarily all the commands you might need to test your design. I have only designed commands to test specific functionality that is required for the assignment. If you have other internal functionality, you might have to add new commands to be able to fully test your implementation.

Initialization Commands

1. *init_mem*

   The directive allocates and initializes data structures for managing physical and logical memory for both operating system memory and user memory.

Status Commands

1. *list {user | system | pagetable [page_table_id]}*

   The directive "list" can have several arguments. The argument of "user" will list the data structures for user memory, while the argument of "system" will list the data structures that are reserved for the operating system memory. If the argument of "pagetable" is issued, then the associated pagetable will be listed. If the command has no arguments, then it will display all user structures, all system data structures, and all defined pagetables.

Action Commands

1. *alloc_pt page_table_size*

   This command effectively tells the memory manager that a new process is being created and the process manager needs to allocate a page table to the process. The size of the page table is passed as an argument and page_table_id should be returned.

2. *dealloc_pt page_table_id*
   This command instructs the memory manager that the page table is no longer needed and it can be reclaimed and reused. It would be issued by the process manager when a process has been terminated as part of the termination procedure.
3. *page_fault page_table_id page_num*
   This command is to bring a specific page into the memory space of the process that caused the page fault. It would normally be executed when a running process makes a

reference to a page that is not currently residing in memory. If physical memory is full, then your page replacement algorithm is to determine which page to replace. You will need to provide a stub that normally would be called to move the page to backing store, should it need to be moved.

Expected use of this test interface will be like:

- Multiple “alloc_pt” commands to simulate the creation of several processes
- Some page faults to simulate the loading of pages into memory
- An occasional “list” to see what is where
- You should fill physical memory with page faults to fill the physical space, then see the effect of your replacement algorithm.