CSCE 611

Operating Systems

MP 4

Virtual Memory Management and Memory Allocation

Design Document

Student: Ram Sankar Koripalli

The objective of this MP is to implement a virtual memory Manager which supports large address space and does virtual memory allocation using recursive page table approach

PART:1 SUPPORT FOR LARGE ADDRESS SPACES

To implement a virtual memory manager for larger address space, we use the Process Memory Pool to allocate memory for the Page Directory and Page Table. We use the Recursive Page Table approach to map the logical address to the frames

Logical address structure

Page directory page (10) | page table page (10) | offset (12)

Page directory is accessed as follows

| 1023 | 1023 | page directory page no.(10) | 2

Page table page is accessed as follows

| 1023 : 10 | page directory page no. (10) | Page table page (10) | 2

Implemented APIs

PageTable::PageTable()

page table and page directory tables are initialized in this constructor

PageTable::load()

page table is loaded into the memory by setting the cr3 bit

PageTable::enable_paging()

paging is enabled by setting the cr0 bit.

PageTable::handle_fault(REGS * _r)

checks whether the address is valid and calculates the page table and page directory index

PART:2 REGISTERATION OF VIRTUAL MEMORY POOLS AND LEGITIMACY CHECK OF LOGICAL ADDRESS:

Implemented APIs

PageTable::register_pool(VMPool * _vm_pool)

Virtual memory pools are stored in the linked list

VMPool::is_legitimate(unsigned long _address)

Checks whether the given address is valid or not by comparing whether the address is in the range of base and base+limit.

PART:3 ALLOCATION AND DE-ALLOCATION OF THE VM_POOLS

The virtual memory manager allocates memory using the registered vm pools and tracks the allocation using an array which contain the base address and the length of the vm pool

Implemented APIs

VMPool::allocate(unsigned long _size)

The number of required pages is calculated based on the given size. Then the base address and the length of the region is calculated and the base address is returned to the caller.

VMPool::release(unsigned long _start_address)

The given address is searched across the base address of the allocated memory regions and then all the pages belonging to the memory region are freed using the base address and the page size variables

PageTable::free_page(unsigned long _page_no)

This function finds the address of the frame by translating the logical address to physical address and the frame is released from the continuous frame pool