Design Document
Page Table Management

CSCE 611
Operating Systems
MP3 - Fall 2018
by
Himanshu Gupta

This machine problem implements a page table manager, that handles page frames for the memory in an x86 machine. The table is created by a two-level hierarchy where each entry in table is divided in 3 different sections. There are two types of table, one for maintaining the page directory and the other page table itself.

To handle a page fault, we first check whether the fault is in page directory or page table. This is done by check the entry from page directory whose index is received via the cr2 register. If the fault is in page directory itself, we first allocate a frame for the page table from kernel memory and then the actual page from the process memory pool. There are several shift operations required for this implementation which are mentioned accordingly in the page table file.

The other functions defined in this file are

1. Init_paging – we initialize the private variables of the PageTable class.
2. Constructor- we first create the page table for directly mapped memory. This is determined by the shared memory size. We mark all the pages as present in this memory. We mark rest of the frames as not present.
3. Load: this api load the current object of PageTable on the current_page_table variable and writes the address of page directory on cr3 register by using write_cr3
4. Enable_paging: this sets the private variable enabled_paging as true and marks the cr0 register by the specified value.

TESTING

This problem was tested by the already written test suite in kernel.C
I modified the number of pages being accessed to more than 27 MB and the program was able to throw an error saying insufficient memory. Similarly, if I request any memory less than 27 MB, which is the total available memory, it successfully allocates that.

I added other debugs in the code to verify the bits that I have already set.

Another requirement of this testing was that MP2 functionality should work correctly, I did not face any error from same as the frame pool manager was able to provide all the required frames and just added an assert case if we request memory more than 27 MB.

So, if the user tries to access more than 27 MB, the code will assert that this much memory is not available.