

MP3: Page Table Manager

Sathvik Kote Revanasiddappa

UIN: 634005706

CSCE611: Operating System

Assigned Tasks

Main: Set up and initialize the paging system and the page table infrastructure for a single address space : Completed.

System Design

Objective

This machine problem aims to introduce to a demand-paging based virtual memory system for our kernel. The objective for the problem is setting up and initializing the paging system and the page table infrastructure for a single address space, with the potential to expand to multiple processes and address spaces later on.

Page Management

The memory layout is as follows:

- The total amount of memory in the machine is 32MB.
- The first 4MB is reserved for the kernel.
- Memory within the first 4MB will be directly mapped.
- Memory over 4MB are freely mapped.
- The first 1MB contains all global data, memory that is mapped to devices.
- The actual kernel code starts at address 0x100000, i.e. at 1MB

Code Description

Page Manager Design

According to the specifications, the page manager will map the first 4MB of the virtual address space to the first 4MB of the physical space. To map the kernel address space, the page manager requires one entry in the Page directory frame and one full frame in Page table pages. The frame manager are borrowed from MP2.

The page manager after initializing the address space, loads the CR3 with the address of its page directory and enables the 31st bit in CR0. The MMU will continue the operation until there is a page fault.

A static method is bound to the error handler to handle the page faults. The page manager will follow the below steps when there is page fault.

- Read the fault address from CR2
- Extract the page directory and page table offset from the faulted address
- If the page directory offset is not valid, the page manager will allocate a kernel frame for page table pages and update the entry in the page directory.
- Once the page table page is valid, a frame from the process frame pool is obtained and the page table entry is updated. The entry is set to 7, indicating the user frame, Read and write, and valid frame.

Implementation Details

The below files are modified. And description gives information regarding the modification.

Mac Support

I have modified the below files to support the functioning of MP3 problem on Mac M2.

- copykernel.sh
- bochsrc.bxrc

cont frame pool.H/C

Updated the contents of both the files from MP2.

page_table.H

- unsigned long pde_address: Holds the address value of the page directory.
- unsigned long get_page_table_frame(int no_of_frames): Uses kernel frame pool manager to allocate a frame for page table.
- unsigned long get_process_frame(int no_of_frames): Uses process frame pool manager to allocate a frame for the physical memory.
- bool is_valid_entry(unsigned long entry): Checks whether the given page table entry is valid or not based on the present bit.

page_table.C

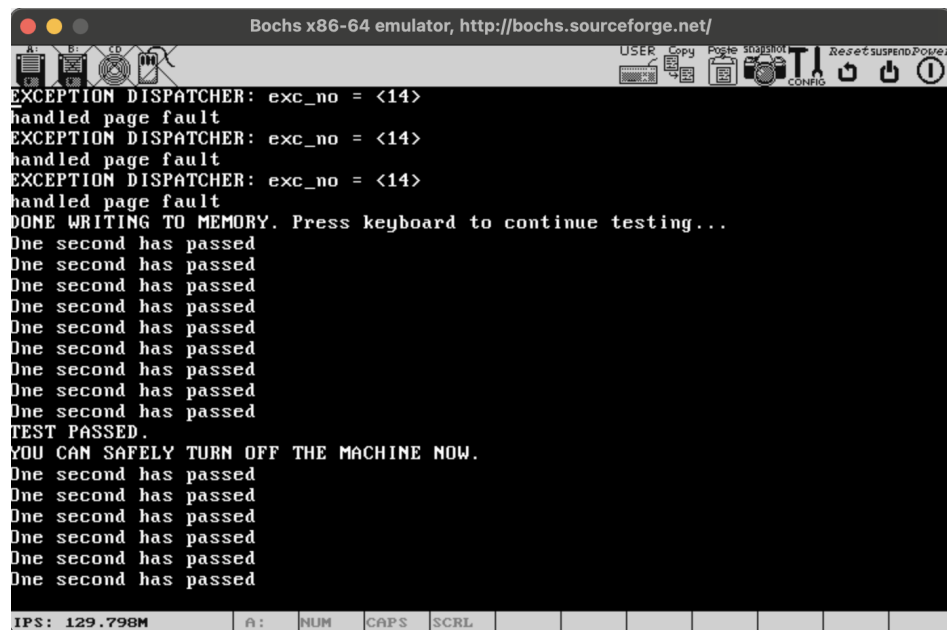
- void PageTable::init_paging: The method is used to initialize the static variables kernel pool and process pool. It also updates the size of the shared space between the kernel space and process space.
- PageTable::PageTable: The constructor follows a series of steps to initialize the page table.

- It allocates a frame from kernel frame pool as page directory frame.
 - It allocates frames for mapping kernel space from kernel frame pool manager.
 - Every page table entry in page table page will be a multiple of 4096 (starting from 0) and are made as valid bit.
 - Once the page table page is set up, the page directory is updated with address of page table page address. The entry is then turned valid.
- void PageTable::init_page_table: The method which is used to init a page table. It can be used to set everything to zero or with some initial value and flags. Used in clearing the page table.
 - void PageTable::load: The function will load the page the address of page directory into the CR3 register.
 - void PageTable::enable_paging: The function will set the 31st bit of CR0 to enable paging.
 - unsigned long PageTable::get_page_table_frame: The function which is used to obtain a kernel frame and return the address of it.
 - unsigned long PageTable::get_process_frame: The function which is used to obtain a process frame and return the address of it.
 - bool PageTable::is_valid_entry: Checks whether the page table entry is valid or not.
 - void PageTable::handle_fault: The static method is bound to exception handler 14. It follows a series of steps to resolve the page fault.
 - Read the fault address from CR2
 - Extract the page directory and page table offset from the faulted address
 - If the page directory offset is not valid, the page manager will allocate a kernel frame for page table pages and update the entry in the page directory.
 - Once the page table page is valid, a frame from the process frame pool is obtained and the page table entry is updated. The entry is set to 7, indicating the user frame, Read and write, and valid frame.

Testing

- Testing initialization of page table manager: This test validates the page initialization and ensures the virtual address is mapped to physical address in the first 4MB.
- Testing Resolution of page faults: This test checks whether the page table manager is able to allocate a page for an address that which raised a page fault.

Both the tests are passing.



The image shows a screenshot of a Bochs x86-64 emulator window. The title bar reads "Bochs x86-64 emulator, http://bochs.sourceforge.net/". The window contains a black terminal area with white text. The text shows a series of "EXCEPTION DISPATCHER: exc_no = <14>" messages, each followed by "handled page fault". This is followed by "DONE WRITING TO MEMORY. Press keyboard to continue testing...". Then, there are ten "One second has passed" messages, followed by "TEST PASSED." and "YOU CAN SAFELY TURN OFF THE MACHINE NOW.". Another ten "One second has passed" messages follow. At the bottom, a status bar shows "IPS: 129.798M" and a row of keyboard status indicators: "A:", "NUM", "CAPS", "SCRL", and several empty boxes.

```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
DONE WRITING TO MEMORY. Press keyboard to continue testing...
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
TEST PASSED.
YOU CAN SAFELY TURN OFF THE MACHINE NOW.
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
One second has passed
IPS: 129.798M  A:  NUM  CAPS  SCRL
```

Figure 1: Testing Results