Design of MP3

The objective of MP3 is to get started on a demand-paging based virtual memory system for our kernel, which is built on x86 architecture. In this problem, we restrict ourselves only in a single process. First, we set up and initialize the paging system and the page table infrastructure for a single address space, directly mapping the first 4MB logical address on first 4 MB in the kernel frame pool. After doing that, the memory above 4MB is going to be managed. The logical address space of our process running on the x86 system is 32-bit, which is corresponding to 4GB. However, our memory used for this process is only 27MB (15-16MB is marked as accessible). Thus, we have to manage our memory by dynamic allocation, which means we map the used portions of logical memory to physical memory frames as soon as the logical memory is called for the first time. As is presented in the handout, memory pages in the logical address above the 4MB mark do not have any explicit equality relationship with physical memory. Whenever such a page is referenced, a page fault in our kernel occurs (Exception 14, which is already set up in kernel.C file), and a page fault handler is responsible for this exception, allocating a frame of physical memory for this logical page to write and read, which needs us to complement its code.

The overall logic of this assignment is very clear. Now we need to show some important technical details

1. Our paging system is used to serve the x86 system, so we should follow the routine as x86 always managing their memory, which is a 2-level paging system, page directory, and page table page. We set the first 10 bit in 32-bit address as PDE (page directory entry) and the second 10 bit as PTE (page table entry), each of them with 1024 entries. And the last 12 bit is offset, corresponding to 4KB space in logical and physical memory. However, we do not want to waste it and utilize it to record some information of this page. In this problem, we only use the last one bit to check whether this logical page has been allocated in physical memory or not.
2. The role of each function in the PageTable class has been described in detail in the handout. After initialization, the first 4MB is directly mapped on physical memory. We first test whether this mapping works or not to check our code, especially for the code managing pointer.
3. We use cont_frame_pool completed in MP2 to get frames from physical memory. As is presented in the handout, page directory and page table pages should be put in the kernel frame pool. When handling page fault, the handle_fault function first tracks the page directory, and then it will find whether we need a new page table page or it has already been constructed. If needed, a free frame is gotten from the kernel pool to construct a new page table page. After that, a free frame is called from the process pool, which is beyond 4MB, to hold this logical page.
4. As is convenience, we just copy the whole cont_frame_pool in MP2 in MP3 and it works well. However, we have to admit that our cont_frame_pool focus on managing consecutive frames. The longer consecutive frames you call, the less fragmentation you will get. In MP3, each time handling a page fault, we only get one frame by our cont_frame_pool, which generates a lot of fragmentations. As is shown in Figure 1, in this test, we set NACCESS in kernel.C as (7MB)/4, which

will allocate 1835008 (7*1024*256) consecutive integers in logical memory address from 4MB to 11MB. However, our test stopped at the logical address of 11276288, which is about 10.75MB. This really makes sense. Because our cont_frame_pool always find the first two bit in an 8-bit char in the bitmap, so if we only use the cont_frame_pool handling single frame, 3/4 of the frames will be fragmentations. And reducing 1MB as marked inaccessible (15-16MB), this result matches very well with MP2. And if we change NACCESS to (6.5MB)/4, the test succeeds perfectly, as is shown in Figure 2. Therefore, in MP3, if we want to reduce fragmentations as much as possible, we should use simple_frame_pool instead of cont_frame_pool. I will make this replacement in future MP4.

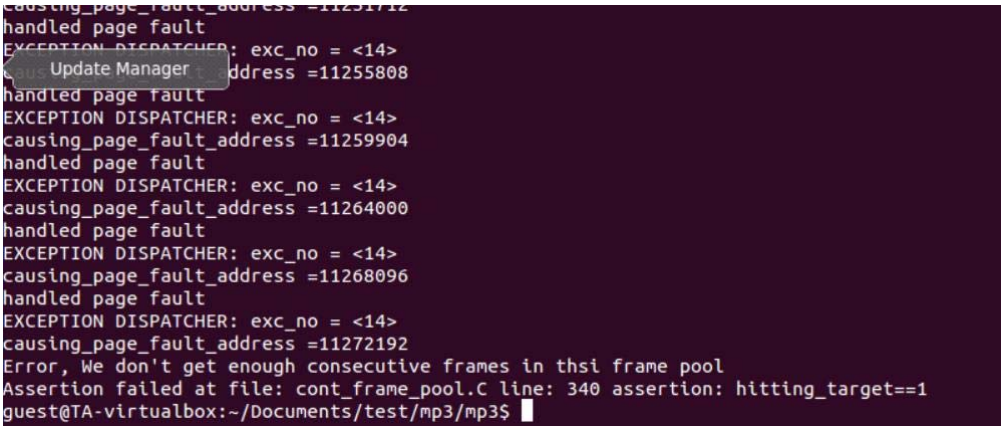

Figure 1. Fragmentations causing not enough frames

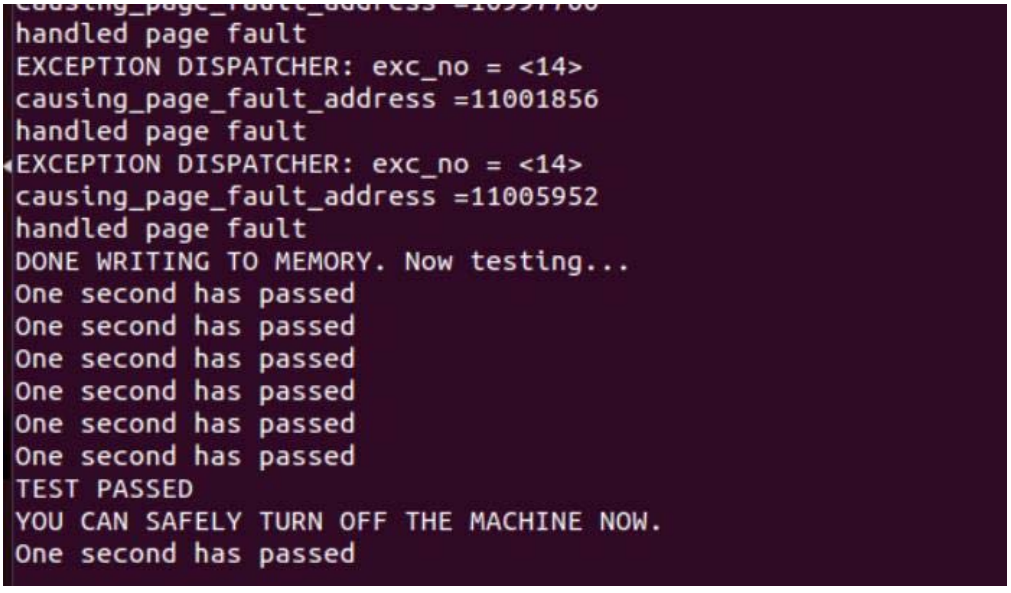

Figure 2. Successful Test