Design documentation

Name: Shiyang Chen UIN: 623009273

FramePool

This class is used for allocating frame for kernel memory space and process memory space before PageTable object is constructed.

This class has 4 private data field: **startframe_no**, **frame_no**, **infoframe_no**, **indicator**; **startframe_no** means the frame number of first frame of all frames that will be allocated; **frame_no** means total numbers of frames that will be allocated; **infoframe_no** means the frame number of the frame that stores the management information for all frames allocated this time; **indicator** is static Boolean type array has 8*1024 elements, each indicating the current availability of a particular frame, "1" means the frame is free and "0" means the frame has been allocated. Notice that all physical memory space is 32MB with frames 4KB in size, so the total number of frames is 8*1024.

Besides functions(or called methods) pre-declared in **frame_pool.H**, I add one static function called **frame_init()** to make program more understandable. It explicitly initializes all frames availability before allocating any frame. It marks first 512 frames (2MB) as inaccessible and the rest of frames allocatable.

Whenever a frame is needed, **get_frame()** is called and the corresponding frame is marked as unallocatable. When searching a free frame, I simply allocate the first frame that is free.

Similarly when a frame is released, **release_frame()** is called (though it's never be called in kernel.C) and the corresponding frame is marked as allocatable again.

Some other details are included in comments in source code file.

PageTable

With no modification on **page_table.H**, all the work is defining declared function.

init_paging() simply pass parameters to corresponding data field.

PageTable constructor first give **page_directory** a physical address, then fill in the first page directory entry **page_directory** points to, which contains the address of first pagetable and some flags. After that, it fills in 1024 entries in the first pagetable one by one. Finally it should also fills in the rest of page directory entries (1023) as "non-present". Details are included in comments in source code file.

load() is mainly a call of **write_cr3()** and **enable_paging()** is mainly a call of **read_cr0()** and **write_cr0()**.

handle_fault(REGS * r) first use 10 highest bit in faulted virtual memory to locate an entry in page directory and call **get_frame()** to fill in the entry. Then it use 10 bit in the middle in virtual memory to locate an entry in corresponding page table

and call again **get_frame()** to fill in the entry. Because I assume page tables are stored in kernel memory space, the first **get_frame()** calling object is **kernel_mem_pool** whereas the second **get_frame()** calling object is **process_mem_pool.** Some other details are included in comments in source code file.