Jacob Hessong

MP2 Design Document

This machine problem focused on the implementation of a frame manager through the use of continuous frame pools. The memory for this machine is a total of 32MB with the first 4MB reserved for the kernel and a frame size of 4KB.

To do this, we needed to implement 5 functions within the ContFramePool class, these being the constructor, get_frames, mark_inaccessible, release_frames, and needed_info_frames. The release_frame function is static, therefore it does not need a frame pool object to be called, meaning it is designed to free all memory. To keep track of the 4 different states that the memory units can be in, I used 2 bits to represent the frames, with 01 meaning the frame is allocated as the head of a pool, 11 meaning its allocated but not the head, 00 meaning it is free memory, and 10 meaning it is inaccessible.

Other than the standard variable necessary to keep track of the frames via the information passed into the constructor, the variables used were a singularly linked list of pools, a pointer to the head of the list, and a bitmap to track the allocated frames and their state information.

Implementation of each function:

1. Constructor – First, take all arguments and assign them to their respective variables. Then, use the info_frame_no to set the bitmap. Next, all of the frames are set as free since no memory has been allocated yet. Finally, it sets up the initial values for the linked list and head pointers.

2. get_frames – This function searches the frames to see if there is a contiguous portion of frames long enough to fit the requested amount of memory. If it is not found, the function prints an error message and returns 0. Otherwise, it sets the first frame as the head and the rest as allocated before returning the first frame number.
3. mark_inaccessible – This function first checks if the range of frames requested is actually in the bounds of the memory. If so, it goes through and marks them all as inaccessible using 10.
4. release_frames – This function first iterates through the frame pools and uses the ranges to find the correct pool. Then a bitmap pointer is used to point to that pools bitmap so that it can be edited within this static function. It then goes through the requested memory and sets their information as 00 to mark them as freed.
5. needed_info_frames – This function just returns how many management frames are needed for a given memory size.