Anu Khatri

CSCE 410-500

Professor Da Silva

February 26th, 2021

P2 – Design Document

Constructor

The majority of the constructor was similar to SimpleFramePool, but I had to adjust a few things to make sure that everything was accounted properly, such as removing a few 8 (whether it be modding or multiplying) since I am using bytes instead of bits. I added a list of pools in the form of a linked lists to keep track of the pools, so I made a PoolNode struct, which was used in the constructor of the ContFramePool. At the end of the constructor of ContFramePool, I added to the linked list and adjusting the pointers to make sure everything was in the right order. This was most useful for release_frames used later.

get_frames

The bulk of the code was in a while loop that was iterated until a suitable range was found for the desired number of frames. A few indexes were used for various reasons: i was to keep track of the index in the bitmap, frame_index was used later in the function for the beginning index of the found range used to change those to head/allocated.

The first if statement is to catch if the index every goes over the total number of frames (n_frames). After that, I keep increasing the index number until a free frame is found. Frame_no is then updated. I then check to make sure the range location isn't beyond n_frames. If that is ok, then I have a for loop that checks to make sure the whole range is free after the first found free frame. Once the last frame is found to be also free, the Boolean for the while loop is changed to true, and it exits the while loop. If any frame is found to be a head/allocated, I start over the search and update the index to start from there.

Once the range is found, I use frame_index to remember where to start changing the status of the frames. The first one is head, and the rest are allocated. n_free_frames is also decremented once those are all changed.

Finally, the frame_no should be returned.

mark_inaccessible

This one is simple, as it is just a for loop calling the mark_inaccessible helper function

mark_inaccessible (helper)

I have a few asserts making sure we are in the correct range, as well as checking if it isn't already a head/allocated. After those all pass, the frame is marked as allocated. And the n_free_frames is updates.

release_frames_help

I made this helper function for release_frames. This makes the frames in the given pool found from release_frames, marked as free. I mark all of the frames as free until a new head/allocated is found, indicating that it would be the end of the range desired to release

release_frames

This function uses some range checks to see which pool the given frame_no is, and it is done by traversing the pool_list created. Once the frame_no is found in a given pool, the release_frames_help function is called to finish the job.

needed_info_frames

This calculates the needed number of frames to store the bitmap information. It is similar to the given example with a few adjustments numbers-wise to account for bytes instead of bits.