CSCE 611

Gautham Srinivasan (UIN: 927008557)

MP2 Design Document

Objective:

The objective of machine problem 2 is to design a frame pool manager which manages allocation and release of frames which are used by kernel frame pool (2MB) and process frame pool (28MB) and mark certain frames as inaccessible (15MB) which are off limits to user.

Contiguous Frame Pool Design:

Contiguous frame pool manages frames for kernel pool below 2MB-4MB and frames for process pool above 4MB-32MB. It used bit map methodology where two bits of info frames are mapped to a frame which indicates the state of the frame (given below). Also, since two bits are used, one byte of bitmap can store 4 frames. The size of the frame is 4KB, then the total number of frames that can be stored in a bitmap is 4KB * 4 = 16K frames can be stored in a bitmap.

11 → **Frame is free**

00 → **Frame is allocated**

10 → **Frame is the first frame of contiguous frames (head frame) and it is allocated**

a) **Contiguous Frame Pool Constructor** initializes the data structures for contiguous frame pool from the starting base frame number till base frame number + number of frames. It also uses info frame number which is used to store the state of the frame in a bitmap. If the info frame number is 0, then the first base frame number is used as info frame.

b) **get_frames** API takes number of frames requested as an argument and returns the head frame number. This function searches for free frames ie 11 in the bitmap and mark the first free frame as head frame (10) and rest of the frame as allocated frame (00). It then returns the head frame number. This API allocates only contiguous frames and skip non-contiguous frames even if they are available.

Detailed Explanation:

1. Run a loop till total frames/4 since one byte in bitmap can hold 4 frames.
2. If the frame is free, then increase the count till you get the total number of requested frames.

3. If the frame is head frame, reset the count to zero. This will handle non-contiguous frame allocation.
4. Mark the first frame as head frame and mark rest of the frames as allocated in the bitmap.

c) **release_frames** API takes the head frame number as input and returns nothing. This function first check whether the frame is inside process pool or kernel pool; marks the head frame as free and mark rest of the allocated frames as free till it reaches another head frame or free frame.

Detailed Explanation:

1. Identify whether the frame is in kernel frame pool or process frame pool.
2. Check whether the first frame is head frame.
3. Mark the head frame as free
4. Mark the rest of the frames as free if the frame is allocated.

d) **mark_inaccessible** API takes the base frame number and number of frames as input and returns nothing. This function marks the frames – from base frame number till base frame number + number of frames as head frames to make it as inaccessible.

Detailed Explanation:

1. Similar to get frame logic.
2. Frame base frame to the last frame, mark them as head frame to indicate they are allocated.

e) **needed_info_frames** API takes total number of frames as argument and returns the number of info frames required to store the total frames. In simple pool, 1 bit is used for addressing the frame, so the total number of frames that can be stored in the bitmap is 32k. Similarly, in contiguous pool, we are using 2 bits per frame, so the total frames that can be stored in bitmap are 16k. Hence, the total number of info frames is :

total frames / 16k + (total frames % 16 k > 0 ? 1 : 0)