Document

Zhiren Zhou

September 2022

1 Design Concept

1. Implemented a new data structure **bitmap_char_** to store the bitmap:

   The size of **bitmap_char_** is one byte and it contains 4 parts with 2 bit in each part. Every part is used to represent the state of one frame.

2. Implement the enum class **FrameState** to make the state of frame more readable.

   FrameState has 4 different values. **Free** means this frame is not used. **Used** means this frame is already allocated. **Hos** means this frame is the first frame of continuous allocated frames. **Inacs** means this frame is not allowed to access. These 4 states are the '' translation" of the 2-bit value part in the bitmap.

3. Implemented the **get_state()** and **set_state()** function:

   get_state() function is used to get the state of certain frames with return value in **FrameState** format. **set_state()** function is used to set the value of 2-bit part in bitmap so that the state of certain frame is consistent with the desired state.

4. Implemented the **linked list** to store the existing pool manager.

   Every instance of ContFramePool(pool manager)has the **next** pointer to point to the next pool manager. The class ContFramePool has its static pointers **head** and **tail** to store the whole list.

5. Implemented the **constructor** function:

   The constructor function initialize the private data members of the pool manager object. The most important thing here is: If this object is a kernel pool manager, we store its bitmap in the first frame of the pool. Otherwise, we get some frames from kernel pool before creating the process pool manager. Then we store the bitmap of process pool in the those frames of kernel pool.

6. Implemented the **get_frames()** function:

   Find a continuous **_n_frames** in the pool. If find successfully, set the first frame of these frames to be **Hos** and the left to be **Used** and return the **absolute frame number** of the first frame. Otherwise, return 0.

7. Implemented the **mark_inaccessible()** function

   Make sure the given frame number is managed by this frame. If yes, set the state of those frames to be **Inacs**.

8. Implemented the **static release_frames()** function:

   First, use the **linked list** to find out which pool manager "responds" for frames to be released. If successfully found, go into that pool manager and ask it to release the given continuous frames. Note: the given frame number in **static release_frames()** is **a absolute frame number** and must be in the state of **Hos**, which means that frame number marks the start of continuous frames.

9. Implemented the **release_from_my_pool()** function:

   This is a private function belonging to the pool manager. When the **static release_frames()** finds which pool manager responds for that given frame number, the certain pool manager calls this function to release the continuous frames. This function will check the given frame number is the **Hos** (start of continuous frames) or not.

10. Implemented the **static needed_info_frames** function:

    This static function is used to calculate how many frames(rounded up) are needed to store the bitmap given the size of one frame pool.

2 Files Modified

1. **cont_frame_pool.H**
2. **cont_frame_pool.C**
3. **kernel.C** (for testing process pool)

3 Testing Results