P2 DESIGN

Operating Systems – CSCE 613

ABSTRACT

Design for Memory Management and API discussion

Tharun Battula

82400197

DESIGN

- The specifications mentioned in the assignement are followed
    - 32 MB of memory layour
    - Kernel Pool 2MB to 4MB
    - Process Pool beyond 4MB
    - Inaccessible region 15MB to 16 MB
- BIT MAP structure -
    - I used 2 bits for storing info about single frame
        - 1X → Inaccessible
        - 01 → Frame Allocated
        - 00 → Frame Free
    - Unsigned char structure is used for storing information 4 frames info i.e., 4 * 2 bits = 1 Byte in Unsigned char
        - MSB represents higher frame info
        - LSB represents lower frame info
    - For frame size of each4KB
    - For Kernel Pool → 2bits * 4MB / 4KB = 2048
        - = 256 bytes < 4KB i.e., frame size
        - 1 single frame is used
        - This pool info is stored at hardcoded address 512 i.e., first frame in kernel pool. At 2MB point
    - For Process Pool → 2bits * 28MB / 4KB = 14336
        - = 1792 bytes needed
        - 1 single frame is used < 4KB
        - This pool info is stored at the first frame request from the kernel pool. i.e., the first frame provided by the kernel pools get_frame() request.
        - This address is observed to be hardcoded address 513 i.e., second frame in kernel pool after the 2MB memory mark.

Functionality

- The regular expected API functionality as per specification are implemented
    - FramePool class is implemented with the necessary functions.

- To debug the code → frame_pool.h file make debug enable flag to 1.
  `#define DEBUG_ENABLE 1`

- Corner Cases
  - In get_frame() requested
    - If all frames are either allocated or inaccessible
      It will be treated as failure and return 0
  - If release_frame(frame_no)
    - If frame_number is beyond the region managed by the frame pool, it will not release the pool
    - If the requested frame number is inaccessible it will not be released by the pool
  - If Mark_inacceble() is called on inaccessible area, it will not have no effect
    - The only way of making inaccesible region accessible is reallocating the frame pools with exact same location and exact same bit map info frame as it was requested first time.

Assumptions

- First kernel FramePool will be called, It keeps its bitmap info in the first frame residing inside the FramePool
- Get_frame from kernel frame pool will be called before process frame pool
- Process Frame Pool will be called based on the frame number obtained from the get_frame request in the kernel frame pool. The frame allocated in kerne frame pool will be used for storing the process frame pool information.
- If the frame_pools are released.
  - If the core information of frame pools is released, I am assuming the user will not work on the frame pools, Until re allocated.
- If frame pool is re allocated with the same size and same location, It will start fresh info map on the same location.
  - The previous get_frame info will be lost

        - The previous release info will be lost
        - The previous mark_inaccessible info will be lost

API Modifications

- Removed static from

  ```
  void release_frame(unsigned long _frame_no);
  ```

  a. It Releases frame back to the given frame pool.
  b. The frame is identified by the frame number.
  c. If this function asks for releasing inaccessible frame
      a. It will not free it,
      b. It prints out warning on the screen for Log purposes.
  d. If this function asks for releasing out of range frame -
      a. It will not free it.
      b. It prints out the warning on the screen for Log purposes.

  e. NOTE: This function is changed Non static, Hence use only object instances for calling this function
      a. It is good coding practice to keep the information within the pool structure
      b. Avoiding Static/Global variables
      c. Making Static required extra memory for storing every instance
      d. Design becomes complex to maintain the extra memory
      e. If Linked list is used, looking inside linked lists take more time in design also caching
      f. It is easier implementation

TESTING

- Tested basic features asked in the assigment by individually checking bits
- Stress testing by reallocation of pools and
    - Getting and asking for all frames
    - Releasing all the frames
    - Marking accessibility randomly
    - Random testing in combinations of functionalities