CSCE 611: Operating Systems

Fall 2022

Name: Rajesh Satpathy

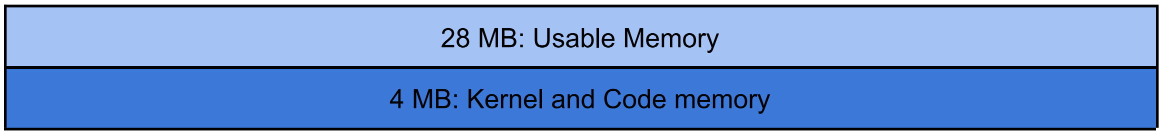
UIN: 931006158

Goal:

Build a Frame Pool Manager for an on demand-paging based virtual memory operating system with a given kernel.

Assumptions shared about the given Kernel:

- Total machine memory: 32 MB
- Memory Layout:



- About the first 4 MB:
  - Is reserved for the kernel.
  - The first 1 MB address space is reserved for IDTs, GDTs, video memory, etc.
  - The address range from 1MB to 2MB consists kernel code and stack space.
  - The kernel frame pool that is getting implemented in this assignment is located between 2 MB an 4MB.

Implementation:

A frame pool manager is implemented within cont_frame_pool.C, containing the following functions:

- Constructor
- set_state
- get_state
- get_frames
- release_frames
- mark_inaccessible
- need_info_frames

Design Decisions:

The design decisions for the data structures used is as follows:

- A singly linked list is used to traverse through frame pools.
- A bitmap of 8 is used for storing frames. A char datatype (8 bits) is used for each bitmap.

- Each frame is represented by 2 bits, so as to support 3 states of the frame -> Head of Sequence, Used and Free b'10, b'11 and b'00 respectively..
    - The frame of a frame pool consists of 3 types of frames which are stored in an enum as:
        - Free: The frame is free to use -> b'00
        - HoS: The frame is allocate and is the Head of a Sequence of frames -> b'10
        - Used: The frame is allocated and in use -> b'11

Constructor:

The constructor initializes the frame pool.

- As we have more than one frame pool and it is efficient to dynamically assign them across the memory according to the number of frames contained, an implementation of a linked list is introduced.
- The initialization of a framepool is made by adding a new set of free frames with a base frame number to linked list of frame pools.

set_state:

Bit manipulations of b'11[i.e. 3] and b'10[i.e. 2] bits are used to set bits on the required frames

- Free: Set bits at frame by bitwise '&' on a frame shifted with modified mask of b'11.
- Used: Set bits at frame by bitwise '|' on a frame shifted with mask of b'11.
- HoS (Head of Sequence): Set bits at frame by bitwise '|' on a frame shifted with mask of b'10.

get_state:

Bit manipulations over b'11[i.e. 3] and b'10[i.e. 2] bits are used to check bits on the required frames

- Free: Check bits at frame by bitwise '&' on a frame shifted to check if frame under consideration is b'00.
- Used: Check bits at frame by bitwise '&' on a frame shifted to check if frame under consideration is b'11.
- HoS (Head of Sequence): Check bits at frame by bitwise '&' on a frame shifted to check if frame under consideration is b'10.

get_frames:

Traverse the given pool's bitmap to get a block of required contiguous free frames and return the first frame number with this condition. The first frame is marked as Head of Sequence and the rest n frames are marked as used.

release_frames:

Traverse through a list of frame pools to find the pool with given frame number and release all its used frames. After the required frame pool is found, it is checked whether the frame is a Head of Sequence (which it is always) and is marked free. The trailing frames are marked free until they reach another Head of Sequence frame or a used frame.

`Note: b'<numbers> represents binary numbers`

<u>mark_inaccessible</u>:

Mark a set of frames inaccesible, same as get_frames and ignores the existence of contiguous blocks of the required n frames size.

<u>needed_info_frames</u>:

Return the number of frames required to store management information for a frame.

`Note: b'<numbers> represents binary numbers`