ARYAN GHOLAMI
JULY 17 , 2024

Memory Management System Documentation

Introduction

This project is a memory management system that simulates RAM and RAM management. It is written in Python. It uses the my_memory list data structure to simulate RAM and uses classes to manage free space and track where files are located in RAM. Also,It supports virtual memory when necessary.

Global variables

I. My_memory : simulate RAM and store data in this DS
II. Free_memory : A list of MemoryFreeSpaceHandling objects
III. Memory management algorithm : which algorithm we should use
IV. name_base_limit_req : list of binding objects

Class MemoryFreeSpaceHandling

ATTRIBUTES:

I. Name : The name of the free segment
II. Start : The starting address of free space in this segment
III. End : end of segment

Class Binding

There is some important information about a file which in stored in memory

ATTRIBUTES:

I. Filename : The name of file
II. Base , limit and seq : Where is file
III. Virtual_memory_state: A boolean indicating if virtual memory is used
IV. frame_size : The size of each frame in virtual memory (if used).
V. num_frame : The number of frames required for the process (if virtual memory is used).
VI. page_table : The page table for the process (if virtual memory is used).

METHODS:

I. address_binding : Takes a line and returns the line's byte
II. get_name : Returns name of the file

Function get_arr()

Takes from user number of free space and the bytes of free space and Initializes the free_memory list with MemoryFreeSpaceHandling objects and the my_memory list with the corresponding free spaces.

Function set_algorithm()

Prompts the user to choose a memory management algorithm from First Fit, Worst Fit, Best Fit, and Next Fit. Sets the global variable memory_management_algorithm accordingly.

Function is_free()

Checks if there is any free space available in memory.

Function find_space(size , algorithm)

Finds a free space in memory that can accommodate a process of the given size using the specified algorithm. Returns the segment of free space.

Function store(i, name)
Stores the process represented by the file name in the ith segment of memory.

Function update_free_memory(i, size)
Updates the free_memory list after storing a process by adjusting the starting address of the free space segment.

Function show_memory_state()
Displays the current state of free memory spaces.

Function get_file()
Prompts the user to input file names and calls store function. Updates the free_memory list and the name_base_limit_reg list with binding information.

Function run()
Main function to run the program. Prompts the user to input a file name and a line number to read from memory. Handles file reading and memory management operations.