Operating Systems
Assignment No.3

Owais Waheed (ow07611)

November 19, 2023

1 Data Structures

The memory layout I implemented consists of two main components **Stack** and **Heap**. To implement these components, there were some underlying data structures used which are defined below:

Memory Stack: This data structure has a static list of the struct frame status. The size of the list is equal to number of frames allowed on stack. There is also a top pointer which tells the current position of pointer/head responsible for reading and writing on the stack.

Frame Status: Frame status have information about the function which have requested a frame on the stack. It tells about function address in code, function name, stack frame number assigned to that function and the frames address. There is also a flag which tells us about the current frame of execution.

Memory Heap: The heap is implemented using two linked lists. One keeps track of the free region and the other manages the allocated region.

Free List Node: It contains the starting address of the free region, its size and the address of the next free region.

Allocated List Node: It contains the name of variable which has requested the space on the heap, its starting address and the address of the next allocated region.

Memory: This is a character array of size 500 to replicate the memory layout.

2 Algorithms

In this section, I will explain the algorithms used to implement the required functions.

Create Frame This function initializes the frame on stack for the function. It takes function name and address as an argument and then checks for the correct length of function and space availability on the stack. Once satisfied, it then initializes all the members of frame status and copies it into memory array using memcpy and also places the stack pointer at the start of the frame.

Delete Frame This function deletes the data of the frame from memory. It sets the member of stack frame to null and replace all the data on memory with empty spaces.

Create Integer/Double/Char This function checks for the space on stack frame. If the stack is not exceeding 300 bytes and the stack frame is not exceeding 80 bytes it creates the requested variable on the stack.

Create Char Buffer on Heap This function first checks for the space of a pointer on stack frame. After that it iterates over free list to find the required space and return the first region which fulfills the space requirement. If heap is full, it requests for more space on memory and after every free space allocated, it performs coalescing by merging neighbouring free space region (free regions which overlaps each other) into one big free space chunk. After updating free list it adds the allocated region in the allocated list as a node.

Deallocate Buffer on Heap The deallocate heap function releases an allocated buffer on the heap identified by name. It searches for the corresponding allocated node in the list, deletes it, clears the memory occupied by the buffer, and updates the free list with the reclaimed space. The algorithm efficiently manage memory, coalescing adjacent free blocks to mitigate fragmentation.

Show Memory The show memory function provides an image of the current state of the memory, including stack frames, heap, free list, and allocated list. It iterates through the memory array, printing details of each stack frame and its associated information. It then displays the stack frames' content and extends to showcase the heap, printing characters up to the current heap size. Additionally, it prints the free list and allocated list, providing a visual representation of memory utilization.

Main Function The main function sets up the necessary data structures (head for the free list and allocated head for the allocated list) and then enters a loop to continuously accept user commands. It uses fgets to read the input and then parses the input based on the expected command formats. Depending on the command, it calls the appropriate functions.

3 Results

Here is the snapshot of the Memory instance:

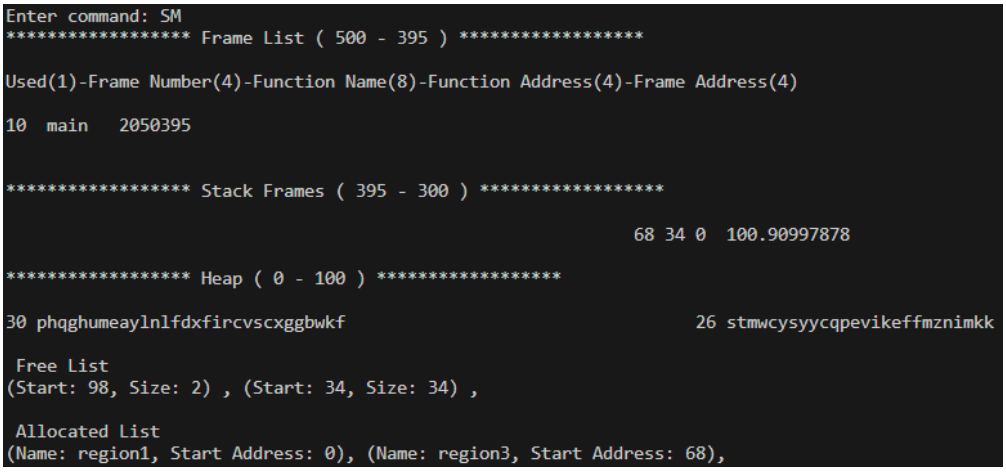

Figure 1: Description of the result