CSL 3020: Computer Architecture

Dynamic Memory Allocation: Role in Memory Management

Report

- Rahul Barodia (B20CS047)

Dynamic Memory Allocation: Role in Memory Management

Venue: Vishwabharati Academy's college of Engg, Ahmednagar(Dist), University of Pune, Maharashtra, India

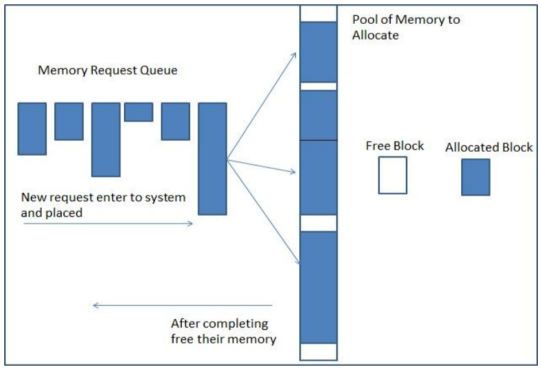
Publication Year: 2014

Web Link : [(PDF) Dynamic Memory Allocation: Role in Memory Management]

With the advancement of new technologies, modern computers must adapt to requirements such as efficient memory management, resource management, etc. On a computer, the application programs cannot directly run on the hardware; it will require an interface called Operating System. Memory Management is a very complex part of the Operating system.

Problem Addressed:

In Modern days, social networking websites are increasing, everything is stored on computers, increasing numbers of users etc., so there is a huge demand for memory than its availability. The main objective of memory management is to use the available memory as efficiently as possible. To understand the problem, let's look at the diagram.



We have certain empty memory blocks in the main memory for allocation. The upcoming requests for memory allocation can only be achieved if the size of the upcoming request processes is less than or equal to the size of empty memory blocks in the main memory.

A queue is present to handle this memory request. A limitation is that the memory allocation algorithm should not use more memory space or more processor time to run.

Methodological Contribution of the Paper :

One possible solution for this memory management problem is to use **Static memory allocation process**. Static memory allocation process is done at compile time; we have to allocate all the memory required to program, applications or variables in its lifecycle at the beginning of execution. This seems to be an easy and feasible approach, but it creates many problems like internal fragmentation, memory cannot be changed while executing a program, memory wastage, etc.

These problems can be handled in **Dynamic memory allocation/management**, sometimes called Manual Memory Management. In DMA, the memory is allocated at run time. It is allocated whenever a program, application, data, or variable demands the required amount of bytes. 'C' programming language supports malloc(), calloc() and realloc() functions to allocate memory for our program or applications. These functions are called dynamic memory allocation functions; with the help of these functions, we can allocate the required memory size at run time. The most important advantage of this strategy, memory management can perform a better role when there is short of memory because it allocates required bytes of memory only at run time and de-allocates memory after its use for reuse of that memory.

Working: For dynamic memory allocation, C programming is being used here. Suppose our task is to allocate memory dynamically for 1000 integers and, after the process is completed, de-allocate it. Now in C programming, there is a header file called **stdlib.h** (general purpose standard library), which contains functions like **malloc()**, **calloc()** and **realloc()**. These functions are used for allocating memory, and **free()** function is used for deallocating memory. Let the variable name be data.

int * data; (Dynamic allocation requires pointer variable instead of standard variable)

Next, for dynamic allocation of this data, we use **malloc()** function and **sizeof()** function. Here **sizeof()** function gives the number of bytes required integer data type because size of integer is different in different OS (Linux: 4 bytes and Windows: 2 bytes).
Malloc() function takes the size in bytes and allocates that much space in the memory.

data = (int *) malloc (10000 * sizeof(int));

(here malloc() function returns void pointer, so we have to convert it into integer).

Now for the deallocation of memory, we use **free()** function.

free(data);

Now, the process of memory de-allocation is completed.

In the case of dynamic memory allocation of arrays, instead of using pointers, the best method is to create a data structure that starts from a small piece of memory and adds a new piece of memory whenever required. Pointer is connection here which holds these pieces.

We have to remember the following points during dynamic allocation of memory:

- Memory allocator has the responsibility to look which memory space is filled and which is empty.
- When we obtain memory from the global pool, We must assume that it contains garbage and properly initialize it.
- We must not assume that contiguous memory block allocation will take place as successive requests will arrive.

Key Takeaways:

- In dynamic memory allocation, the memory allocation can be done at run time.
- Memory size can be changed based on the requirements of the dynamic memory allocation.
- There is no memory wastage.

Drawbacks:

1) External Fragmentation: It occurs when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. There is enough memory space to complete a request, but it is not contiguous.

2) Dangling Pointer: The data pointer still points to the memory location even when that memory is de-allocated. Such a pointer is called a Dangling pointer.

Possible solution for the above Drawbacks:

1) External Fragmentation can be removed by compaction. It means shuffling memory contents to place all free memory together in one large block so that the incoming memory request can be allocated.

2)To remove dangling pointer, we can use

data= NULL;

Now, the data pointer will point to null.