Volume 5, Issue 8, August 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Computational Study of Static and Dynamic Memory Allocation

Meenu[*]
Department of Computer Science,
RGGCW, Bhiwani, India

Vinay Dhull
Department of Information Technology,
TITS, Bhiwani, India

Monika
Department of Computer Science
RGGCW, Bhiwani, India

Abstract— Memory management is one of the important function of the operating system so it plays a vital role in modern computer system. Dynamic Memory Allocation is an efficient technique of Memory Management and becomes essential part of today's computer system. It is critical problem in computer science by paying some complexity. It utilize the processor at the full potential by minimizing the cost of memory and it is art of handling computer memory efficiently. Memory Management is commonly one of the most serious part of Operating System, especially the main memory. This paper includes the analysis of Dynamic Memory allocation and static memory allocation in Memory Management, comparison between both, and role of DMA..

Keywords— ***DMA: Dynamic Memory Allocation, MM: Memory Management, DMM: Dynamic Memory Management, OS: Operating System. CPU: Central Processing Unit.***

I. INTRODUCTION

Now a days the Computer science firms are going through significant changes because of the development of new technologies that are moving towards big data. Modern computer systems assimilating requirements, such as efficient memory management, resource management, good implementation process, virtual memory management, effective interprocess communication, good user interface etc. for operating system design to continuous business process reengineering. To determine these substantial requirements for modern operating system design, we are required to concentrate on these requirements. OS sits between the user and hardware as shown in fig.1

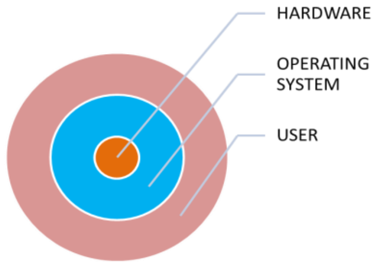


Fig. 1. OS acts as interface b/w user and hardware.

Application programmers doesn't directly interact with the hardware, these will require interface called Operating System. Operating system is an interface between hardware and application programs or end user. Designing of OS is not an easy task; we have to care about previously mentioned criteria. It is mostly developed in Assembly, C, and C++ programming languages. The basic functions of the operating system includes process management function , memory management function, storage management function, device management function and protection and security as shown in fig. 2

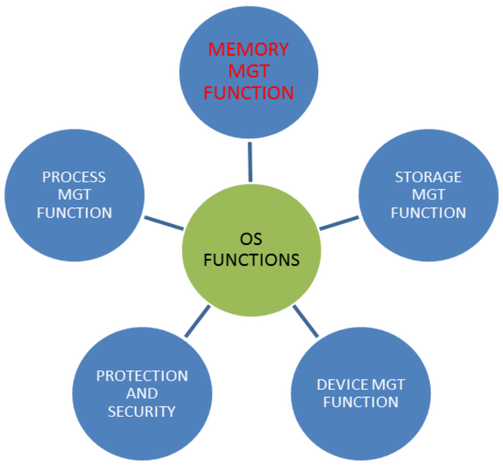


Fig. 2 Functions of operating System

We are going to address here the Memory Management parts of operating system.
Memory Management is very complex part of Operating system design because it is related to physical level or can say interaction to hard disk. It is broadly divided into three parts:

- Hardware Memory Management,
- Operating System Memory Management,
- Application level Memory Management.

Hardware level memory management includes RAM and cache memory, which are related to hardware devices those actually store data.
Operating System Memory Management is related to the memory allocated to programs. The OS can provide computer more memory than it actually has, and the program has also the machine's memory. These both memories are part of virtual memory.
Application level memory management consists of two related tasks: Application and Recycling. It provides memory for objects of program and data structures. When the program demands for memory then the memory manager allocate a block of memory and if any data which is no longer required then recycles that particular block of memory. Application memory manager also works on CPU overhead, Interactive pause time and Memory overhead. These are three main memory management components, in this paper we are discussing on Operating System Memory Management.

II. MEMORY MANAGEMENT DESIGN PROBLEM

In present days, social networking websites are increasing, everything is going to stored in computers' memory, rapidly growing the numbers of users etc. so there is an immense demand for the memory than its availability. Memory management keeps track all the allocated and de-allocated memory locations in the system. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time and for how much time . It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status. The main objective of memory management is to use the available memory as efficient as possible. The Memory management design problem is shown in following fig. 3.
In this we have to start with a large piece of memory. We will receive the sequence of blocks for the request of memory. A request can be completed with a block of memory that is at least as large as the size requested. After completing this request the memory is in use for a while, and it is returned to allocator. Queue is handling all the memory requests. The limitations are that the memory allocation algorithm should not use more memory space and more processor time to run. There is no best solution to memory management design, there are some series of solutions. Each solution has some advantages and disadvantages and we have to find out which is the best solution for us. But it is hard to find out the best solution without implementing and testing to the selected solutions.

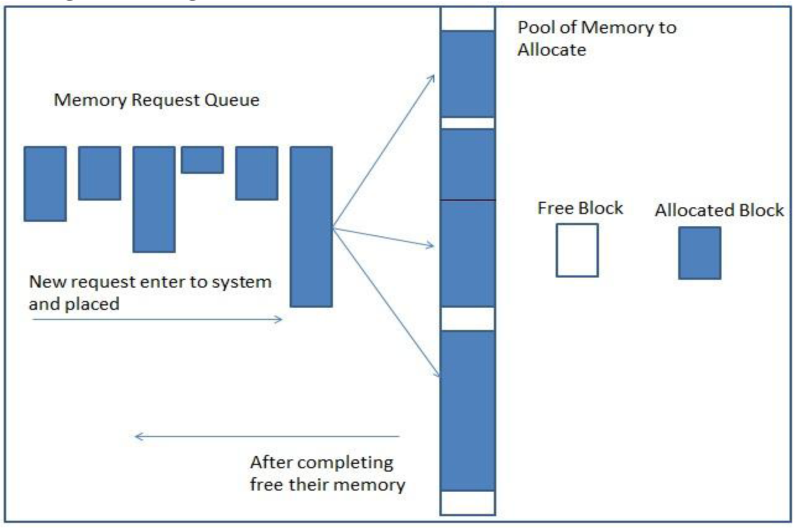

Pool of Memory to Allocate
Memory Request Queue
Free Block
Allocated Block
New request enter to system and placed
After completing free their memory

Fig.3 Memory management design problem

(A) STATIC MEMORY MANAGEMENT PROBLEM

Static memory allocation process is done at compile time; we have to allocate all the memory which is requiring to program applications or variables in its lifecycle at beginning of execution. Even we don't need most of the memory at particular instance of program or variable, still we could not use allocated memory for other purpose.
As we know, every day large amount of data is generated termed as Big data, need not only stored big data but also maintain copies of Big data for fast accessing purpose and to avoid data loss due to system crash or natural disaster like earthquake etc. so that we have to use available memory in efficient manner. Memory management issue came because of inefficient way of allocating and de-allocating memory.
For instance we declare static array to store integer data, when we declare array of 1000,
Required memory = 1000*2 = 2000 bytes on windows (32 bit) and
Required memory = 1000*4 = 4000 on Linux or 64 bit windows.

2000 (on Windows) and 4000 (on Linux) has reserved for above declared array and we could not use that memory, even that array contains only one integer or no data. Suppose if an array has 1000 integers, we have deleted 999 still we could not use that memory for other purpose.

Above use of memory is called static memory allocation. Above problem with memory management arises because of static memory allocation or we can say, in such situation static memory allocation fails to handle memory management efficiently. Still static memory allocation has few advantages over dynamic memory allocation like allocating speed faster, mostly no fragmentation problem, no extra algorithms required for allocation task. Even these benefits with static memory allocation still we prefer dynamic allocation mostly because need to use available memory efficiently since data increases very fast day by day**.**

(B) DYNAMIC MEMORY MANAGEMENT/ALLOCATION

In Dynamic Memory Allocation , the memory is allocated at run time. Whenever any process or application requires memory ,it will be allocated to them at run time with the required amount of memory . There are different functions to allocate and de allocate the memory according to the different programming languages like 'C' programming language supports calloc(), malloc() 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 allocation of memory at run time is possible . The important role about this strategy of allocation is that once utilized allocated memory, then that memory can be utilized by other programs or applications after deallocation of the previous used memory. The memory cannot be further used by other programs until the memory de-allocated by the memory manager when there is no longer requirement of memory. 'C' language has provided free function to de-allocate unused memory (C++ Programming language used 'delete' keyword to free allocated memory 'new' keyword to allocate memory). This strategy also has some advantages and disadvantages.

The most important advantage of this technique is to manage the memory when there is no more space in the memory. Because it allocate require bytes of memory only at run time and de-allocate memory after its use for reuse that memory. Operating system provide mechanisms to keep track on the allocation and de allocation of memory and also keep track that for how much time the memory is allocated to which process.

(C) STATIC V/S DYNAMIC MEMORY ALLOCATION

Memory management has merits and demerits of using static memory and dynamic memory allocation. The best strategy utilize the processor to its full potential ,so we have to select the best one between static and dynamic memory allocation technique.

To compare static and dynamic memory allocation, this example shows that how dynamic memory allocation uses memory efficiently than static memory allocation in some situations.

Suppose in the new opened organization, there is a requirement of maintaining the information of employee of organization. To implement employee information management system (EIMS), we can create EIMS in two ways

- one using static memory allocation and
- second using dynamic memory allocation.

Before implement EIMS, we have to consider the development of our organization after 8-10 years, approximate how many employees will be on payroll. Consider at the beginning of this organization only 15 employees and after 8 years organization may have 15000 employee approximately.

To implement EIMS using static implementation, we can use array. We will declare array of structure for 15000 employee, because once developed its very difficult to change.

Second way is to implement EIMS using dynamic implementation, no need to allocate memory for 15000 memories at beginning. Whenever we will add new employee the memory is allocated at run time.

Now we are going to address advantages of dynamic allocation over static allocation. As per above example from beginning of EIMS using statically, we have to allocate memory for 15000 employee even organization has 15 employee. There is a wastage of memory by using the static memory allocation technique. While using dynamically implementation of EIMS, the new employees get the memory at run time . This shows dynamic allocation efficiently used memory than static allocation. Second advantage is that when we delete the employee in static memory allocation ,we can't use the memory in future while in the dynamic allocation when we delete any employee ,we can easily use that memory for the other purpose. Comparision between static and dynamic memory allocation.

STATIC ALLOCATION	DYNAMIC ALLOCATION
Memory is allocated before the execution of the program begins	Memory is allocated during the execution of the program begins
Size must be known at compile time.	Size may be unknown at compile time.
Variables remain permanently allocated	Allocated only when the program is active.
Pointer is needed to accessing variables	No need of dynamically allocated pointers
More memory space is required	Less memory space is required

III. DYNAMIC MEMORY ALLOCATION WORK

Dynamic allocation plays vital role in memory management. Here is explanation , how does dynamic memory allocation works? C programming language is used to explain the working of the dynamic memory allocation technique.

Problem: Allocate memory for 15000 integers and after executing the process, de-allocate memory for recycled it.

Solution: Header file stdlib.h contains different library fuctions which includes functions like calloc() , malloc()and realloc() for allocation while free() for de-allocation of memory.

The task is to allocate memory for 10000 integers, this can be done in two ways statically or dynamically. This is accomplished dynamically as follows:

Suppose variable name: info (integer type).

Dynamic allocation will require pointer variable instead of normal variable, symbol * (asterisk) shows pointer variable declaration.

int * info;

The above statement, declare 'info' as integer pointer. malloc() function, and also use sizeof() function will gives the number of bytes require for integer data type. It will help because size of integer is changes as per operating system (Linux:4 bytes and Windows: 2 bytes).

info = (int *) malloc (15000 * sizeof(int));

malloc() function returns void pointer, so need to convert into our destination type means integer. Above statement will allocate memory for 15000 integers and 'info' pointer will point to first block of allocated memory.

Now, next task is de-allocation of memory. free() function is used to de-allocate memory. Following statement will accomplish this task:

free(info);

It will de-allocate memory to reuse, but still 'info' pointer will points to that memory location. We can called 'info' pointer is a dangling pointer because it pointing to memory even it is de-allocated. Dangling pointer is a pointer which is pointing to nothing. NULL value is used to remove dangling pointer :

data = NULL;

Above statement will remove dangling pointer that means 'info' will point to null instead of any random memory.

Dynamic memory allocation is performed by the memory allocator when the program is executing and the program (user) or application sends a request for additional memory.

Some issues need to be addressed by the operating system:

- allocating variable-length dynamic storage when required
- freeing up storage when execution is completed.
- controls the storage (e.g. reclaiming freed up memory for later use)

IV. PROBLEM WITH DYNAMIC MEMORY ALLOCATION

Everything has two sides like coin has head and tail. DMA also has two sides; one side is pros that we have seen above. Now we are going to second side are cons of using DMA.

There are some problems with DMA as follows:

Memory leak

Memory leak is a condition in which some programs or application which is continuously allocate memory without ever giving it up and finally run out of memory. It will affect data loss.

Dangling pointer

It is also called as premature free. After using the memory ,the memory deallocate by using the function of memory deallocation (free () in C-language) but pointer will still points to that memory location. Whenever the same memory while allocating to other programs or applications then it will crash or behave randomly. To prevent dangling pointer one more task we have to perform with de-allocating memory is to assigning the NULL value to dangling pointer.

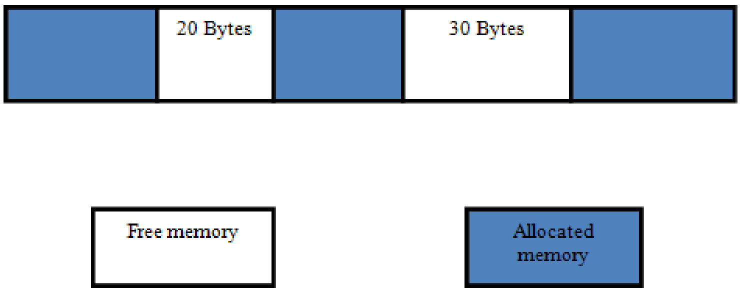
Time consuming

Memory management with the help of dynamic memory allocation is a time consuming process as compare to static memory allocation technique.

Memory Fragmentation

It is very chronic with dynamic memory allocation because of the external fragmentation in some situation. For instance, as we know using dynamic allocation technique the memory may not allocated continuously ,wherever available required size of memory are available ,that memory is allocated to the programs . Suppose we have following memory structure which shown in figure. There is 50 bytes free space available. This 50 bytes of memory available in two parts, first is 20 bytes and second one 30 bytes. Remaining memory has allocated for other purpose.

And request is come to memory manager, to allocate memory of 35 bytes but memory allocator could not allocate memory even it has 50 bytes of free memory, because of fragmentation.



V. CONCLUSION

In this paper the role of dynamic memory allocation in memory management is presented. Different issues are compared between static and dynamic memory allocation with the help of Employee management information system. Dynamic memory allocation proves best strategy for allocating the memory when needed. It utilize the processor to its full potential and hence the efficiency of the computer system is increased.

ACKNOWLEDGMENT

I express my deepest thanks to Mrs. Navita Gill and Mr. Deepak (Asst. Professor, CSE Department) for taking part in discussions and giving necessary advices and guidance, facilitating preparation of this paper.

REFERENCES

[1] David R. Hanson, *Fast allocation and deallocation of memory based on object life times*, Software-Practice and Experience, Vol. 20(1), Jan 1990 pp. 5-12.

[2] Puaut, *Real-Time Performance of Dynamic Memory Allocation Algorithms*, in Proceedings of the 14th Euromicro Conference on Real-Time Systems (ECRTS'02), June 2002

[3] M. Masmano, I. Ripoll, A. Crespo, and J. Real. TLSF*: A new dynamic memory allocator for real-time systems*. In 16th ECRTS, pages 79–88, Catania, Italy, July 2004. IEEE.

[4] Krishna M. Kavi, Merhan Rezaei, Ron Cytron, *An Efficient Memory Management Technique That Improves Localities* University of Alabama in Huntsville and Washington University in Saint Louis

[5] P. R. Wilson, M. S. Johnstone, M. Neely, and D. Boles, *Dynamic Memory Allocation: A Survey and Critical Review*, In Proceedings of the Memory Management International Workshop IWMM95, Sep 1995

[6] M. S. Johnstone. *Non-Compacting Memory Allocation and Real-Time Garbage Collection*. PhD thesis, 1997

[7] Morris Chang, Woo Hyong Lee and Yusuf Hasan *Measuring Dynamic Memory Invocations in Object-oriented Programs* Dept. of Computer Science, Illinois Institute of Technology, Chicago IL. 60616

[8] M. Masmano, I. Ripoll, A. Crispo *Dynamic Storage Allocation for real time embedded systems* Universidad politecnica de Valencia, Spain.

[9] Dipti Diwase, Shraddha Shah, Tushar Diwase, Priya Rathod. (2012). *Survey Report on Memory Allocation Strategies for Real Time Operating System in Context with Embedded Devices*. IJERA Internet Computing [Online]. 2(3), pp. 1151-1156. Available

[10] Manish Mehta, David J. DeWitt, *Dynamic Memory Allocation for Multiple-Query Workloads* Computer Science Department, University of Wisconsin-Madison.

.