International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161
©2014 INPRESSCO®, All Rights Reserved
Available at http://inpressco.com/category/ijcet

Research Article

Dynamic Memory Allocation: Role in Memory Management

Nilesh Vishwasrao Patil[Á*] and Prabhudev S Irabashetti[Ḃ]

[Á]Government Polytechnic, Ahmednagar Maharashtra Government
[B]Department of Computer, Vishwabharati Academy's college of Engg, Ahmednagar(Dist), University of Pune, Maharashtra, India

Accepted 01 March 2014, Available online 01 April 2014, **Vol.4, No.2 (April 2014)**

Abstract

Memory management is an important part of modern computer system. Dynamic Memory Allocation has plays very important role in Memory Management and becomes fundamental part of today's computer system. It is classical problem in computer science by paying some complexity. It is minimizing the cost of memory by providing efficient use of it and it is art of handling computer memory powerfully. Memory Management is commonly one of the most critical parts of Operating System, especially the main memory. In this paper, we will describe the role of Dynamic Memory allocation in Memory Management, comparison with static memory allocation, and issues with using DMA.

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

1. Introduction

Now a days the Computer science industries is undergoing significant changes because the development of new technologies, moving towards big data etc. Modern computer systems must adapts to requirements, such as efficient memory management, resource management, efficient implementation process, virtual memory management, efficient inter process communication, good user interface etc. for operating system design to continuous business process reengineering. To determine these significant requirements for modern operating system design, we need to concentrate to these requirements.

Application programs could not directly run on hardware, it will require interface called Operating Systems. Operating system is an interface between hardware and application programs or end user. The design of OS is not easy task; we have to care about above mentioned criteria. It is mostly developed in Assembly, C, and C++ programming languages. We are going to address Memory Management parts of operating system.

Memory Management is very complex part of Operating system design because it's related to physical level. It is broadly divided into three parts: Hardware Memory Management, Operating System Memory Management, and Application level Memory Management. Hardware level memory management has included RAM and cache memory, which are related with hardware devices those actually stores data. Operating System Memory Management is related with the memory allocated for programs, the operating system can provide computer will have more memory than it has actually, and also program has the machine's memory. These both memories are part of virtual memory. Application level memory management consists of two related tasks such as Application and Recycling. It provides memory for program objects and data structures. When the program has demand for memory then memory manager allocate block of memory and if any data which is no longer require then recycled that block of memory. Application memory manager is also works on CPU overhead, Interactive pause time and Memory overhead. These are three main memory management components, in this paper we have discussed on Operating System Memory Management.

2. Memory Management Design Problem

In Modern days, social networking websites increases, everything stored in computers, increasing numbers of users etc. so there is huge demand for the memory than memory its availability. The main objective of memory management is use the available memory as efficiently as possible. The Memory management design problem as shown in following fig. 1

In this we have to start using 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 request then the memory is in use for a while, and it is returned to allocator. Queue is handling all the memory requests. A limitation is that the memory allocation algorithm should not use more memory space or more processor time to run.

*Corresponding author: **Nilesh Vishwasrao Patil** is working as System Analyst and **Prabhudev S Irabashetti** as Asst Prof

There is no one best solution to memory management design, there are some series of solution. Each solution has some advantages and disadvantages and we have to find out which is best solution for us. But it is always very hard to find out the best solution without implementing and testing to selected solution.

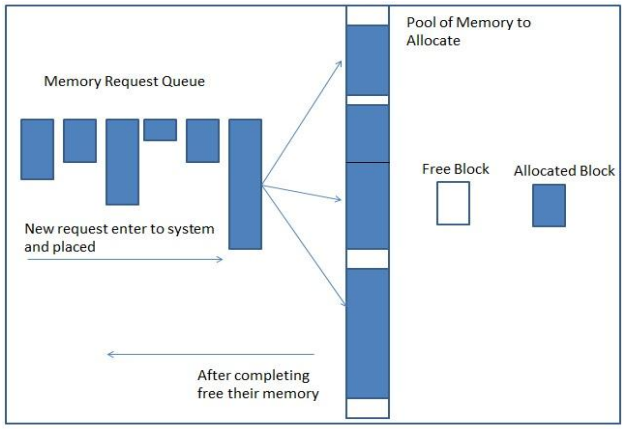

Fig. 1 Memory Allocation Problem

3. 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 variable 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 replication of Big data for fast accessing purpose and to avoid data loss because of system crash or natural disaster like earthquake etc. so that we have to use available memory in efficient manner. Memory management issue is comes because of inefficient way of allocating memory and de-allocating memory.

For example we are declare static array to stored integer data, when we declare array of 5000,

Required memory = 5000*2 = 10000 bytes on windows (32 bit) and

Required memory = 5000*4 = 20000 on Linux or 64 bit windows.

10000 (on Windows) and 20000 (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 array has contains 5000 integers, we have deleted 4999 still we could not use that memory for other purpose.

Above strategy is called static memory allocation. Above problem with memory management is 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 not faced fragmentation problem, no extra algorithms required to achieve allocation task. Even these benefits with static memory allocation still mostly we prefer dynamic allocation because need to use available memory efficiently since data increases very fast day by day.

I will illustrate difference between static and dynamic memory allocation with example, and also see how dynamic memory allocation strategy provide efficient use of memory over static allocation strategy in next part.

4. Dynamic Memory Management/Allocation

Dynamic Memory Allocation is sometimes called as Manual Memory Management. In DMA the memory is allocated at run time. It is allocated whenever program, application, data, variable demands with required amount of bytes. We are going present, manual memory allocation using 'C' programming language because it will very easy to show allocation and de-allocation. The programmer has direct access to memory at run time to control over memory using DMA. It is mostly explicit call to heap management functions. 'C' programming language has 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 required size of memory at run time. The important mechanism about this strategy of allocation, once utilized allocated memory, and that memory is no longer requires for program, application or variable which can be again available to other purpose. The memory manager could not reuse that memory without de-allocate, no longer required memory. 'C' language has provided free function to de-allocate unused memory (C++ Programming language used 'new' keyword to allocate memory and 'delete' keyword to free allocated memory, for your information). This strategy also has some advantages and disadvantages.

The most important advantage of this strategy, memory manages can perform better role when there is short of memory. Because it allocate require bytes of memory only at run time and de-allocate memory after its use for reuse that memory. Keeping track of allocated and free memory is very difficult task but now it has maintained by modern operating system by providing some complexity to this.

5. Static V/s Dynamic Memory Allocation

Memory management has merits and demerits of using static memory and dynamic memory allocation. We have to select best allocation strategy for our business. To select best allocation strategy, need to compare understand: first our business and second merits and demerits of both related to our business.

To compare static and dynamic memory allocation, we are going to take one simple example, so that we could find out how dynamic memory allocation uses memory efficiently than static memory allocation in some situations.

Suppose I have opened an organization, so need to maintain 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 think growth of our organization after 10-15 years, approximate how many employees will be on payroll. Consider at the beginning of this organization only 10 employees and after 10 years organization may have 10000 employee approximately.

To implement EIMS using static implementation, we can use array. We will declare array of structure for 10000 employee, because once developed its very difficult to change.

Second way is to implement EIMS using dynamic implementation, no need to allocate memory for 10000 memories at beginning. Whenever we will add new employee the memory is allocated at run time and we will delete employee we can recycled that memory after de-allocating if we don't that information in future.

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 10000 employee even organization has 10 employee. The memory has wastage means we could not use allocated memory. While using dynamically implementation of EIMS, the memory is allocated at run time whenever new employees add the memory is allocated. This shows dynamic allocation efficiently used memory than static allocation. Second advantage is even we deleted employee from the organization we could not use memory for static EIMS while if we delete employee record the memory will available for us to use for other purpose for dynamic EIMS.

These points have given better idea to us, the role of dynamic memory allocation in memory management. Because of this we can efficiently allocate memory to application, programs or variables. But we have to pay some complexity cost for using dynamic allocation. Following fig. 2 has shown difference between static and dynamic.

Static Allocation	Dynamic Allocation
Performed at static or compile time	Performed at dynamic or run time
Assigned to stack	Assigned to heap
Size must be know at compile time	Size may be unknown at compile time
First in last out	No particular order of assignment
It is best if required size of memory known in advance	It is best if we don't have idea about how much memory require

Fig. 2 Difference between static and dynamic allocation

6. Dynamic Memory allocation work

We have seen how dynamic allocation plays important role in memory management. Now I am going to present, how does dynamic memory allocation works? We have used C programming language for this.

Problem: I would like to dynamically allocate memory for 10000 integers and after completing its work, de-allocate memory for recycled it.

Solution: C Programing language has provided stdlib.h header file, which contains functions malloc(), calloc() and realloc() for allocation while free() for de-allocation of memory.

The task is to allocate memory for 10000 integers; we can do this in two ways statically or dynamically. This is accomplished dynamically as follows:

Suppose variable name: data (integer type).

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

```
int * data;
```

The above statement, declare 'data' as integer pointer. We are going to use malloc() function, and also use sizeof() function which will gives 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).

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

malloc() function returns void pointer, so need to convert into our destination type means integer. Above statement will allocate memory for 10000 integers and 'data' pointer will point to first block of allocated memory.

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

```
free(data);
```

It will de-allocate memory to reuse, but still 'data' pointer will points to that memory location. We can called 'data' 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. To remove dangling pointer we use following statement:

```
data = NULL;
```

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

Above example shows how did efficiently handle memory using dynamic allocation. We have to remember following points during dynamic allocation of memory:

- The memory allocator maintains track of what is free and what is allocated.
- When we obtain memory from the global pool, we

must assume that it contains garbage, and properly initialize it.
- We must not assume that successive requests will allocate contiguous memory blocks.

A dynamic memory allocation takes help of pointers during allocating memory to hold the address of allocated block. Array is decent entity; we could not change its size in middle of execution of program.
- If Array = small then our program fails to handle large amount of data at run time.
- If Array = big then lots of space has been wastage means inefficient use of memory.

The best solution is to create a data structure (language independent entity) which start from small piece of memory and add new piece of memory whenever require at time. Pointer is connection here which holds these pieces.

7. Problem with Dynamic Memory allocation

Everything has two ends like coin is head and tail. DMA also has two ends; one end is advantages that we have seen above. Now we are going to second end are disadvantages 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 the use of memory we have free memory using any de-allocating memory function (free () in C-language) but pointer will still points to that memory location. Whenever programs or application giving up memory and memory allocator attempt to access that latter it will crash or behave randomly. To prevent dangling pointer one more task we have to perform with de-allocating memory is to assign NULL value to pointing pointer.

De-allocating memory

The developer responsibility to de-allocate (free) memory when it finish its work with it. In this two condition commonly seen for example:

Suppose developer has de-allocate memory before he finish it its work then future access to memory will gives run-time error. Second is developer forgotten to de-allocate some allocated space after its use.

To freeing a memory we should have to care, the memory we are going to de-allocate, it's never going to use in future and also try to de-allocate all allocated spaces which will never to be use.

Memory Fragmentation

It is very serious problem with dynamic memory allocation because external fragmentation even we could not use available memory in some situation that we will discuss using following example. As we know using dynamic allocation the memory may not stores continuously, wherever available required size of memory allocated there. Now we are going to see the adverse effect of dynamic allocation on memory management with the help of following example and fig. 3.

Suppose we have following memory structure which shown in fig. 3. There is 30 bytes free space available. This 30 bytes of memory available in two parts, first is 10 bytes and second one 20 bytes. Remaining memory has allocated for other purpose.

And request is come to memory manager, to allocate memory of 25 bytes but memory allocator could not allocate memory even it has 30 bytes of free memory, because of fragmentation

Time consuming

Memory management with help of dynamic memory allocation is time consuming process as compare to static. Let us see, when malloc/calloc/realloc/new (Programming language C, C++) then memory allocator looks for free requested size of block, after finding requested size of free block, mark that block as reserved then assigned pointer to first memory location. There are various algorithms perform this task to achieve dynamic memory allocation.

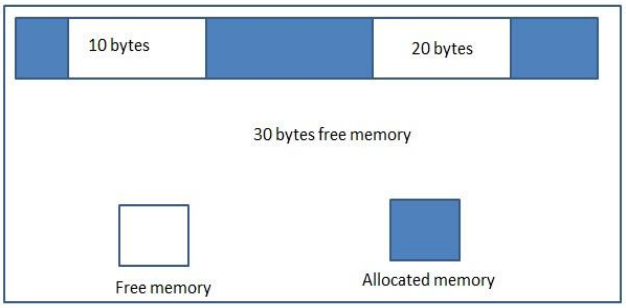

Fig. 3 Memory allocation example

Above are some issues with memory management when it uses dynamic memory allocation strategy.

Conclusion

In this paper we have presented role of dynamic memory allocation in memory management. In which we have discussed memory management design issue, and both ends of dynamic memory allocation (advantages and disadvantages). We have also compared static and dynamic allocation with the help of employee management information system (EIMS). We conclude that dynamic memory allocation is art of handling computer memory with some complexity cost.

Acknowledgment

This paper is completed only because support from each

and every one including: Government Polytechnic, Ahmednagar, teachers, colleague, parents, friends and my students.

Especially, my acknowledgment of gratitude toward the following important persons:

First I would like to thank Mr. Prabhudev sir, Mr. M. Kshirsagar Sir, Mr. Natikar sir, and Mr. Jaypal sir, and my classmates to their support and encouragement.

Second, I sincerely thank to my parents, family, S. A. Bhalerao and friends, who provide the advice and financial support.

Last but not least, my late grandfather and late grandmother for their love. This research paper will not be possible without all of them.

References

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

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

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

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.

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

Website links:

- http://webdocs.cs.ualberta.ca/~holte/T26/dyn-mem-alloc.html
- http://www.memorymanagement.org/articles/begin.html
- http://digital.ni.com/public.nsf/allkb/119E4AA17E0A6F9C86256C4E00568121
- http://www.cprogramming.com/tutorial/virtual_memory_and_heaps.html
- http://www.careerride.com/c-static-memory-dynamic-memory-allocation.aspx

Biography

Mr. Nilesh V. Patil is pursuing ME Computer from Vishwbharti Academy College of Engineering Ahmednagar. He has more than four years of experience involving around two years of industrial experience as Software engineer. He is working as System Analyst in Government Polytechnic, Ahmednagar after being selected through Maharashtra Public Service Commission (MPSC) as Gazetted Class One officer in Nov 2011

Prabhudev S Irabashetti received his M. Tech. degree from Davangere University, Davangere in June 2012. He is presently working as Assistant Professor in Computer Department, Vishwabharati Academy's College of Engg Ahmednagar, Maharashtra, India. He is currently guiding B.E, M.E. students of VACOE Ahmedanagar. He has presented 2 papers in National Conference, and published 2 papers in International Journals