ISSN: 2393-8528

Contents lists available at www.ijicse.in

IJICSE Open Access Journal Approved by UGC

International Journal of Innovative Computer Science & Engineering

Volume 4 Issue 3; May-June-2017; Page No. 127-131

STUDY OF STATIC AND DYNAMIC MEMORY ALLOCATION

Amanjot Kaur Randhawa, Alka Bamotra

Assistant Professor Baring Union Christian College, Batala, District, Gurdaspur Punjab

Assistant Professor Baring Union Christian College, Batala, District, Gurdaspur Punjab

Received 22 May 2017; Accepted 16 June 2017

ABSTRACT

Memory management is one of the most important parts of an operating system. Next to the CPU, it is one of the most important resources in a computer system. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. Dynamic Memory Allocation is an efficient technique of Memory Management and becomes essential part of today's computer system. It solves problem of sharing memory among different processes. 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. This paper includes dynamic and static memory allocation, comparison between both.

Keywords: OS- Operating system, DMA-dynamic memory allocation

INTRODUCTION

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 or user level Memory Management.

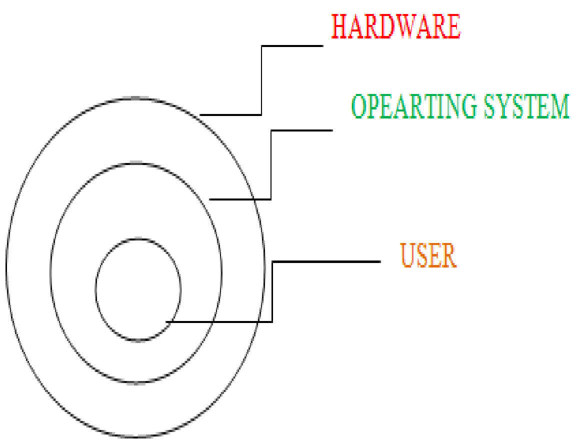

Figure1: OS acts as interface b/w user and hardware.

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.

The main task of operating system is to utilize this memory efficiently. In operating system the main problem faced by memory allocation algorithm is to efficiently allocating memory blocks to the demanding applications with minimum response time. For this purpose different kind of memory allocation designs are being utilized such as the static memory allocation and dynamic memory allocation as shown in the figure.

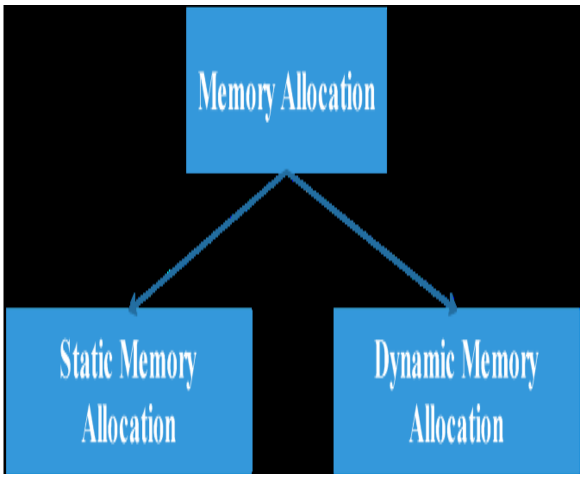

Figure 2:

**Corresponding author: Amanjot kaur Randhawa |*

Both these techniques are supported by real time systems and both of them differ the way the memory is distributed as in case of static memory allocation, memory is allocated at compile time and its known in advance what to allocate while in dynamic memory allocation, the memory is allocation at run time and reference is maintained for allocated and unallocated memory blocks in the form of free and in use memory blocks.

MEMORY MANAGEMENT TECHNIQUES

As shown in above figure two techniques are used in allocating memory –static and dynamic memory allocation techniques. First we will discuss about static memory allocation.

In the presence of different memory management techniques, goal of any memory allocation algorithm rest in providing real time support for memory allocation. Every memory allocation technique has its own pros and cons and it justify their performance for the purpose these techniques are developed. Our intent is to figure out which of these two techniques is best and work more effectively.

STATIC MEMORY ALLOCATION

Static memory allocation process is done at compile time; in static allocation memory is allocated at the beginning of execution to applications, variable. In static memory allocation even if we do not need most of the memory at particular instance of program or variable still we could not use allocated memory for any other purpose.

Every day large amount of data is generated termed as Big data, this big data need not only store but also maintain replication of it for fast accessing purpose and to avoid data loss because of system crash or natural disaster. Memory management issue 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 arises because of static memory allocation, 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 fast speed, 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 of its way of allocating memory. Now we will discuss about dynamic memory allocation and also discuss why dynamic memory allocation is better than static memory allocation technique.

DYNAMIC MEMORY ALLOCATION

Dynamic Memory Allocation is also known 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. First we will discuss dynamic memory algorithm.

Dynamic Memory Allocation Algorithms

An available block of memory to store the process is called a hole. There are four algorithms used to allocate the memory dynamically:
1. First-fit: Allocate the first hole that is big enough. Searching start at the beginning of the set of holes we can stop searching as soon as we find a free hole that is large enough.
2. Next-fit: This behaves exactly as the first fit except that the scan begins from where the previous one left off.
3. Best-fit: Allocate the smallest hole big enough. We must search the entire list, unless the list is kept ordered by size. This strategy produces the smallest leftover hole.
4. Worst-fit: Allocate the largest hole. Again we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.
A study done of the efficiency of next-fit, first-fit, and best-fit showed that in some cases, next-fit performs worse than first-fit or best-fit. When the mean size of the block is less than one-sixteenth the available memory, first-fit performs the best of all, with best-fit close to the performance of first-fit, and next-fit being substantially inferior to both of them.
Now we are going present, manual memory allocation using 'C' programming language because it will very easy to show allocation and de-allocation. Library routines known as "memory management functions" are used for allocating and freeing memory during

execution of a program. These functions are defined in **stdlib.h**.

Table 1:

Function	Description
malloc()	allocates requested size of bytes and returns a void pointer pointing to the first byte of the allocated space
calloc()	allocates space for an array of elements, initialize them to zero and then return a void pointer to the memory
free	releases previously allocated memory
realloc	modify the size of previously allocated space

malloc() function is used for allocating block of memory at runtime. This function reserves a block of memory of given size and returns a pointer of type void. This means that we can assign it to any type of pointer using typecasting. If it fails to locate enough space it returns a NULL pointer.

Example:

```
int*x;
x=(int*)malloc(50*sizeof(int));  // memory space
allocated to variable
free(x);  // release the memory allocated to variable x
```

calloc() is another memory allocation function that is used for allocating memory at

runtime. **calloc** function is normally used for allocating memory to derived data types such as **arrays**. If it fails to locate enough space it returns a NULL pointer.

realloc() changes memory size that is already allocated to a variable.

Example using realloc() :

```
Int*x;
x-(int*)malloc(50*sizeof(int));
x=(int*)realloc(x,100);    // allocated a new memory to
variable x
```

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.

Dynamic v/s Static memory

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 employees of organization. To implement employee information, we can do that in two ways, one using static memory allocation and second using dynamic memory allocation. Before implementing employee information, we have to think about growth of organization after 5-10 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 employee information 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 employee information 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 why dynamic allocation is better than static allocation. As per above example from beginning of inserting employee information using static technique, 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 employee information, the memory is allocated at run time whenever new employees will be added the memory is allocated. This shows dynamic allocation efficiently used memory than static allocation. Second advantage is even if we delete employees from the organization we could not use that memory in case of static technique while in case of dynamic technique if we delete employee record the memory will available for us to use for other purpose.

LITERATURE SURVEY

The work done by various researchers on memory management is as following:

Muhammad Abdullah Awais et.al(2016)[1] proposed that Two Level Segregated Technique(TLSF) is best to use for real time systems because TLSF cause very low internal fragmentation, its response time is very good which is the primary demand of real time system where time is most important factor. Also TLSF allocation and de allocation time is small constant time that makes it much faster than other traditional techniques.

It's different from traditional hoard algorithm because it uses segregated lists in 2 levels as its name suggest. These 3 levels of segregated free lists are used to carry free blocks of memory of same class which reduce internal fragmentation. In first level there are free blocks of memory following power of 2 sequences while 2nd list uses user's configured variables to divide free block classes of first list. Thus help to offer bounded response time.

Dharmender Aswal et.al(2014)[2] suggested the memory management problem. The fundamental goal of memory administration is utilize the accessible memory as effectively as could be allowed. The Memory administration plan issue as indicated in emulating fig. 1.In this we need to begin utilizing a huge bit of memory. We will get the arrangement of squares for the appeal of memory. An appeal can be finished with a piece of memory that is at any rate as expansive as the size asked. In the wake of finishing demand then the memory is being used for some time, and it is come back to allocator.

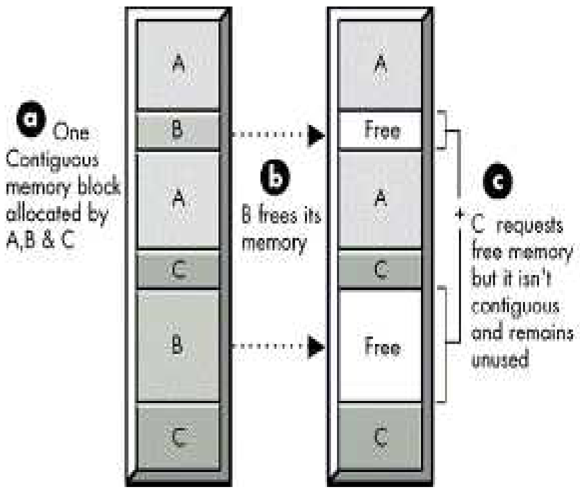
a One Contiguous memory block allocated by A,B & C
A
B
A
C
B
C
b B frees its memory
A
Free
A
C
Free
C
c C requests free memory but it isn't contiguous and remains unused

Figure 3:

A confinement is that the memory distribution calculation ought not to utilize more memory space or more processor time to run. The problem has been solved with the help of dynamic and static memory allocation techniques.

Meenu et.al(2015) [3]also suggested the memory management design problem. 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. This problem has been solved with the help of dynamic memory allocation technique.

Deepak Aggarwal et.al(2003)[4] proposed that Memory allocation is one of the most important duties of an operating system. In the numerous methods memory allocation dynamically, the best is the first-fit method, since it is quick, and minimizes fragmentation. Virtual memory is a common method used to increase the size of the memory space, by replacing frames in the physical memory with pages from the virtual memory. This benefits the operating system in the sense that a process does not have to be completely in memory to execute. To implement a virtual memory system properly, the algorithm with as few page faults as possible must be used to minimize page replacement. The least-recently used algorithm was the best for performance, but the enhanced second-chance algorithm uses several ideas of the LRU method, and is easier to implement. The problems of fragmentation, both internal and external can develop, as well as thrashing can occur even with most careful planning. But, the effects of these problems can be minimized with a careful plan, and an effective memory management system can be implemented.

Prof Manjiri Pathak et.al(2013)[5] suggested the issues & research challenges related to memory management that should be considered while designing the operating system for WSNs.

In WSN, sensor devices are severely constrained by the resources. They usually consist of a processing unit with limited computational power & limited memory, sensors (including specific conditioning circuitry), a communication device (usually a radio transceiver or alternatively optical) and a power source in the form of a battery. As many of the new applications supporting real time traffic require more memory, it is challenging to design the efficient memory management techniques to support these applications. Although great amount of work is done in this area, still many problems have to be resolved. Especially there are many research gaps in case of memory management for WSN in supporting concurrent applications.

CONCLUSION

In this paper we have presented role of memory allocation in operating system. In which we have discussed memory management design techniques i.e static and dynamic memory allocation techniques. We have also compared static and dynamic allocation with the help of an example. We conclude that dynamic memory allocation is better and more reliable as compared to static memory allocation .More research can be done on this topic.

REFERENCES

1. Muhammad Abdullah Awais “Memory Management: Challenges and Techniques for Traditional Memory Allocation Algorithms in Relation with Today's Real Time Needs” international journal of multidisciplinary sciences and engineering, vol. 7, no. 3, march 2016
2. Dharmender Aswal, Krishna Sharda, Mahipal Butola” research paper on DMA:Dynamic memory allocation” 2014 IJIRT ,Volume 1, Issue 6
3. Meenu, Vinay Dhull ,Monika” computational study of static and dynamic memory allocation” International Journal of Advanced Research in Computer Science and Software Engineering 5(8), August- 2015
4. Deepak Agrawal”memory management” Term Paper Operating Systems CS-384
5. Prof. Manjiri Pathak” an approach to memory management in wireless sensor networks” International Journal of Computer Science & Engineering Technology (IJCSET), Vol. 4 No. 08 Aug 2013
6. 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.
7. Manish Mehta, David J. DeWitt, *Dynamic Memory Allocation for Multiple-Query Workloads* Computer Science Department, University of Wisconsin-Madison.
8. Nilesh Vishwasrao and Prabhudev Irabashetti, “Dynamic Memory Allocation: Role in Memory Management”, International Journal of Current Engineering and Technology, Vol. 4, No. 2, April 2014.
9. Dipti Diwase, Shraddha Shah, Tushar Diwase and, Priya Rathod, “urvey Report on Memory Allocation Strategies for Real Time Operating System in Context with Embedded Devices”, International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 3, May-Jun 2012, pp.1151-1156.
10. Puaut, *Real-Time Performance of Dynamic Memory Allocation Algorithms*, in Proceedings of the 14th Euromicro Conference on Real-Time Systems (ECRTS'02), June 2002