INTRODUCTION

1. Heap Storage Management
2. Fixed Size Elements
3. variable Size Elements
4. Solution of recovery Problems

3)Heap Storage Management

It is based on the heap data structure.Heap is a block of storage within which data are allocated/freed in an arbitrary manner. In this the problem of allocation, recovery, compaction and reuse may be complicated.Heap storage management can be of two types:

✓Fixed Size Elements:

This is quite simple.Compaction is not a problem as all the elements are of same size.

✓Variable Size Elements:

This storage technique is programmer control allocation more difficult than the fixed size elements.The major difficulties with variable size element are reuse of recover space.

3)Heap Storage Management

Important point are:

a)As they are not allocated in stack ,the life time of the objects allocated in the heap is not confined to the subroutine where they are created

- They can be assigned to parameters
- They can be returned as value of

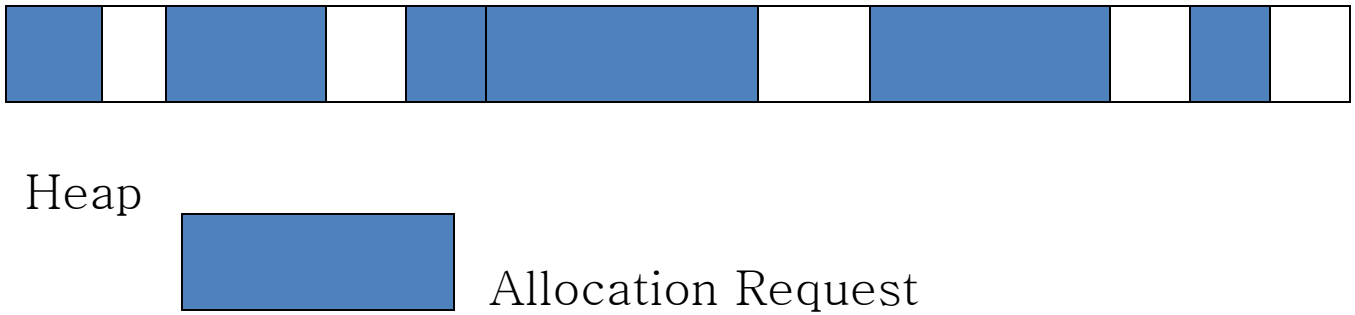
b) In then heap based allocation (heap) objects can be allocated/de allocated at arbitrary times.

The heap is region of storage in which sub block can be allocated/de allocated.

Heap Storage Management

If allocation request can't be fitted in the available space as the request space is large then heap storage management are required.

For example



Properties of Heap storage

1. Generally used as storage for object with an unrestricted lifestyle lifetime
2. Maintains a free list -list of free space
3. On allocation - memory manager finds space and marks it as used

Heap Storage Management

4. On deal location – memory manager marks space as free

5. Memory fragmentation -– the memory fragmentation into small block over the lifetime of program

6. Garbage collection – coalesce fragments, possibly moving objects

The need for heap storage arises when a language permits storage to be allocated and freed at arbitrary points during program execution, as when its language allows creation, destruction, or extension

Three ways to allocation

1. Best fit
2. Worst-fit
3. First-fit

HEAP STORAGE MANAGEMENT

Solution of recovery Problems:

In the process of identification and recovery of freed storage elements in fixed sized heap storage, many problems occurs . Those problems can be solved in the following ways:

Explicit return by Programmer or System

In this method, when an element that has been in use becomes available for reuse , it must be explicitly identified as free and returned to the free-space list. But it is not always feasible because of the following two reasons:

a) **Dangling reference** :In context of heap storage management , a dynamic reference is a pointer to an element that has been returned to free space list means dangling pointer , so that the has been returned to free

Heap Storage Management

Space list means dangling pointer arise when an object is deleted or de allocated , without modifying the value of the pointer so that the pointer still points to the memory location if the de allocated memory. As the system may reallocate the previously freed memory to another process , if the original program then dereferences the dangling pointers, unpredictable behavior may result, as the memory now may contain completely different data . This is especially the case if the program writes data to memory pointed by a dangling pointers, as silent corruption of unrelated data may result, leading to subtle bugs that can be extremely difficult to find.

To construct a particular activation of a subprogram it is required to split in to parts:

A static part :

Also known as code segment made with the help of constant and executable code. It should be invariant during execution of a subprogram and so a single copy may be shared by all act ovation

A dynamic Part

Known as activation record made with the help of parameter, function results and local data some other point like temporary storage areas, return point .

The size and structure of activation record for a subprogram can be find out at translation time

Heap Storage Management

If program attempts to modify through a dangling reference a structure that has already been freed , that contains of an element on the free-space list may be modified inadvertently. If this modification overwrites the pointer linking the element to the next free-space-list element, the entire reminder of the free-space list may become defective. Event worse, a letter attempt by the storage allocator to use the pointer in the overwritten element leads to completely unpredictable results.

b) **Garbage :**In opposite to dangling references discussed above, if the last access path to a storage is without the structure itself being destroyed and the storage recovered, then the structure becomes garbage.

HEAP STORAGE MANAGEMENT

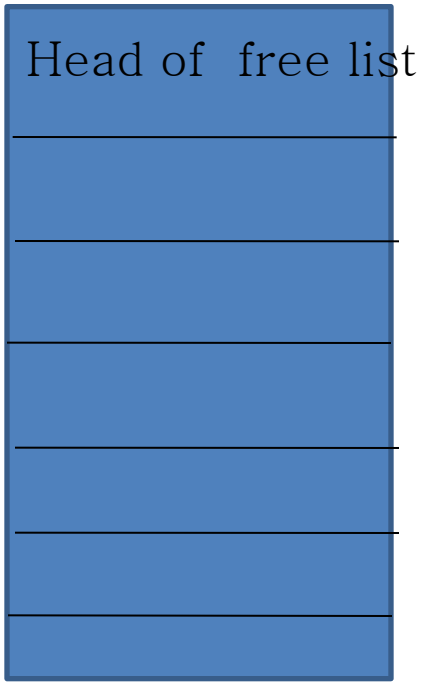
A garbage element is one that is available for reuse but not on the free –space list, and thus it has become inaccessible.

Reference Count:

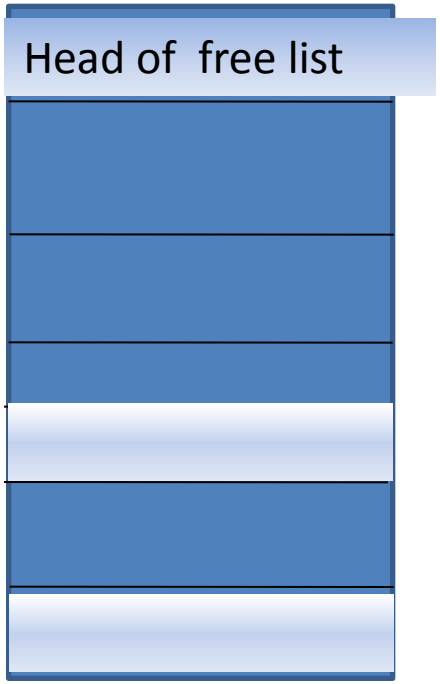
The most straightforward way to recognize garbage and make its space reusable for new cells is to use **reference count.** We augment each heap cell with a **count field** that records that total number of pointers in the system that point to the cell. When an element

HEAP STORAGE MANAGEMENT

is initially allocated from the free-space list, its reference count is set to 1. Each time a new pointer to the element is created, its reference count is increased by 1. Each time a pointer is destroyed , the reference is counter decreased by 1 . If the reference count ever goes to 0, we can reuse the cell by placing it on a free list .



Initial free-space list



Free-space after execution

Shaded areas shows that elements that were allocated and yet not freed

HEAP STORAGE MANAGEMENT

Advantages

- Conceptually simple
- Immediate reclamation of storage

Disadvantages

- Extra space for storing reference counter.
- Extra time(every pointer assignment has to Check change/check count)
- Cost of maintaining reference counter
- Decrease in execution efficiency because testing incrementing and decrementing occurs continuously through out execution.

HEAP STORAGE MANAGEMENT

Garbage collection

The basic principle of how a garbage collector works is:

Determine what data object in a program will not be accessed in the future

Reclaim the resources used by those objects

By making manual memory de allocation unnecessary (and often forbidding it), garbage collection frees the programmer from having to worry about releasing objects that are no longer needed, which can otherwise consume a

HEAP STORAGE MANAGEMENT

Because garbage collection is done only rarely (when the free-space list is exhausted) it is allowable for the procedure to be fairly costly. Two stages are involved:

1. **Mark**: In the first stage each element in the heap which is active, i.e. which is part of an accessible data structure, must be marked. Each element must contain a garbage-collection bit of each active element “OFF”.
2. **Sweep:** Once the marking algorithm has marked active element, all those remaining, whose garbage – collection bit is “ON” are garbage and may be returned to the free-space list. A simple sequence can of the heap is sufficient the garbage collection bit of each element is checked as it is encountered in the scan. If the element is “OFF”, it is passed over. If the element is “ON”, it is linked into the free space list. All garbage collection bits

HEAP STORAGE MANAGEMENT

are reset to “ON” during the scan.

Advantages: Garbage collection frees the programmer from manually dealing with memory allocation and deallocation. As a result, certain categories of bugs are eliminated or substantially reduced:

Dangling pointer bugs, which occur when a piece of memory freed while there are still pointers to it, and one of those pointers is used?

Double Free bugs, which occur when a program attempts to free a region of memory that is already free.

Certain kinds of memory leaks, in which a program fails to free memory that is no longer reference by any variables, leading overtime memory exhaustion.

Heap Storage Management

Disadvantage:
A potential disadvantage of a garbage collected heap is that it adds an overhead that can affect program performance. This activity likely require more CPU time then would have been required if the program explicitly freed unnecessary memory. In addition, programmer in a garbage collected environment have less control over the scheduling of CPU time devoted to freeing objects that are no longer needed.

Heap Storage Management

Variable size elements are sometimes needed, if space is being used for programmer defined data structures stored sequentially, such as arrays, then variable size blocks of space will be required, or activation records for tasks might be allocated in a heap in sequential blocks of varying size. The major difficulties with variable size elements concern reuse of recovered space.