DYNAMIC MEMORY ALLOCATION IN C

PRESENTED BY
M.LAVANYA
M.Sc(CS&IT)
NSCAS

SYNOPSIS

- Memory allocation
- Static Memory Allocation
- Memory Allocation Process
- Memory Allocation Functions
- Allocation A Block Of Memory : Malloc
- Allocation A Block Of Memory : Calloc
- Altering The Size Of A Block : Realloc
- Releasing The Used Space: Free

MEMORY ALLOCATION

- The blocks of information in a memory system is called **memory allocation.**
- To allocate memory it is necessary to keep in information of available memory in the system. If memory management system finds sufficient free memory, it allocates only as much memory as needed, keeping the rest available to satisfy future request.
- In memory allocation has two types. They are **static and dynamic** memory allocation.

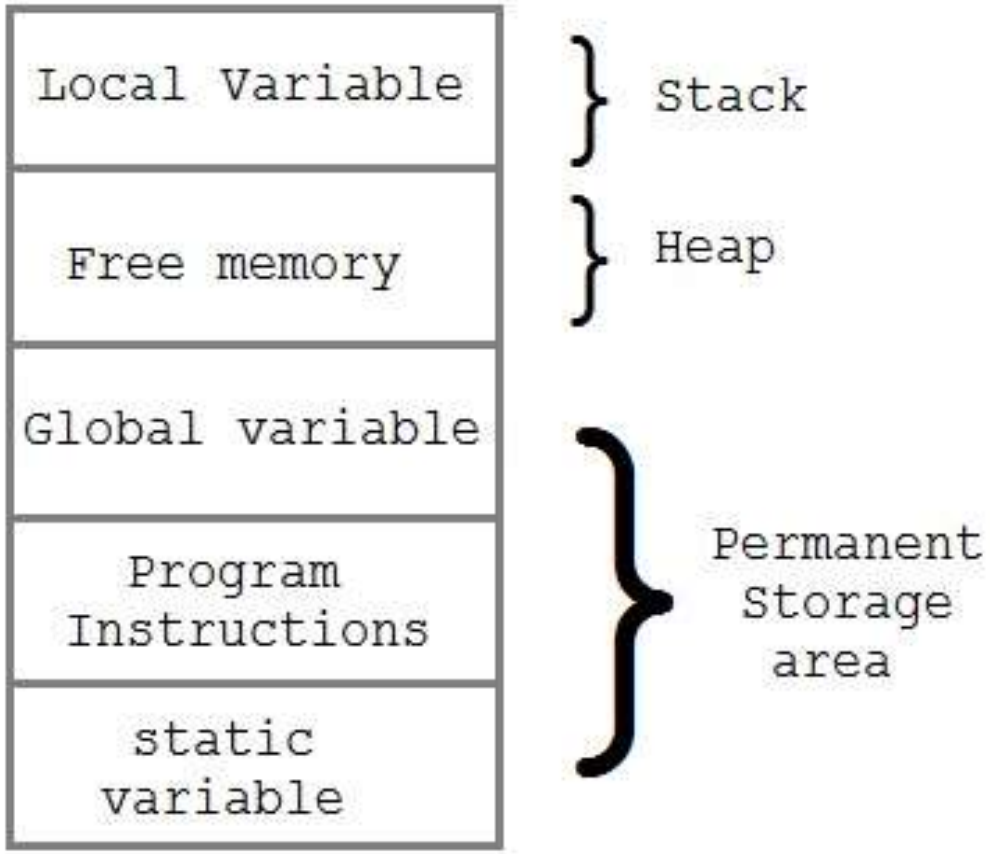
STATIC MEMORY ALLOCATION

- In static memory allocation, size of the memory may be required for the that must be define before loading and executing the program.

DYNAMIC MEMORY ALLOCATION

- In the dynamic memory allocation, the memory is allocated to a variable or program at the run time.
- The only way to access this dynamically allocated memory is through pointer.

MEMORY ALLOCATION PROCESS



MEMORY ALLOCATION FUNCTIONS

Function	Use of Function
malloc()	Allocates requested size of bytes and returns a pointer first byte of allocated space
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer to memory
free()	deallocate the previously allocated space
realloc()	Change the size of previously allocated space

ALLOCATION A BLOCK OF MEMORY : MALLOC

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.

Ptr=(cast-type*) **malloc** (byte-size);

EXAMPLE PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
 struct emp
{
int eno;
char name;
float esal;
 void main()
{
struct emp *ptr;
ptr = (struct emp *)  malloc(size of(struct emp));
if(ptr == null)
{
pintf(“out of memory”);
}
else
{
printf(“Enter the emp deitals”);
scanf(“%d%s%f,&ptr-> eno,ptr-> name,&ptr-> esal”);
return 0;
}
```

Output:

Enter the emp details
eno 1
ename priya
esal 10,000

ALLOCATION A BLOCK OF MEMORY : CALLOC

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** and **structures**.

Ptr=(cast-type*)**calloc**(n,elem-size);

EXAMPLE PROGRAM

```
#include <stdio.h>
 #include <stdlib.h>
int main()
{
int i, n;
 int *a;
 printf("Number of elements to be entered:");
 scanf("%d",&n);
 a = (int*)calloc(n, sizeof(int));
 printf("Enter %d numbers:\n",n);
 for( i=0 ; i < n ; i++ )
{
 scanf("%d",&a[i]);
 }
 printf("The numbers entered are: ");
 for( i=0 ; i < n ; i++ )
 {
 printf("%d ",a[i]);
 }
 free( a );
 return(0);
 }
```

output:

```
Number of elements to be entered :3
 Enter 3 numbers:
  22
 55
  14
The numbers entered are :22 55 14
```

ALTERING THE SIZE OF A BLOCK : REALLOC

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

ptr=**REALLOC**(ptr,new size);

EXAMPLE PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *ptr = (int *)malloc(sizeof(int)*2);
int i;
int *ptr_new;
*ptr = 10;
*(ptr + 1) = 20;
ptr_new = (int *)realloc(ptr, sizeof(int)*3);
*(ptr_new + 2) = 30;
for(i = 0; i < 3; i++)
printf("%d ", *(ptr_new + i));
return 0;
}
```

Output:

10 20 30

RELEASING THE USED SPACE: FREE

Free() function should be called on a pointer that was used either with ”calloc()” or “malloc()”,otherwise the function will destroy the memory management making a system to crash.

free (ptr)

EXAMPLE:

```
func()
{
int *ptr, *p;
ptr = new int[100];
p = new int;
delete[] ptr;
delete p;
}
```


Thank
you