What is Dynamic Memory Allocation in C?

C Server Side Programming Programming

Here we will see what is dynamic memory allocation in C. The C programming language provides several functions for memory allocation and management. These functions can be found in the <stdlib.h> header file. The following functions for memory allocations.

Function	Description
void *calloc(int num, int size);	This function allocates an array of **num** elements each of which size in bytes will be size.
void free(void *address);	This function releases a block of memory block specified by address.
void *malloc(int num);	This function allocates an array of **num** bytes and leave them uninitialized.
void *realloc(void *address, int newsize);	This function re-allocates memory extending it upto **newsize**.

Allocating memory dynamically

While programming, if you are aware of the size of an array, then it is easy and you can define it as an array. For example, to store a name of any person, it can go up to a maximum of 100 characters, so you can define something as follows −

```
char name[100];
```

But now let us consider a situation where you have no idea about the length of the text you need to store, for example, you want to store a detailed description about a topic. Here we need to define a pointer to character without defining how much memory is required and later, based on requirement, we can allocate memory as shown in the below example −

Example Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
   char name[100];
   char *description;
   strcpy(name, "Adam");
   /* allocate memory dynamically */
   description = malloc( 200 * sizeof(char) );
   if( description == NULL ) {
      fprintf(stderr, "Error - unable to allocate required memory
");
   } else {
      strcpy( description, "Adam a DPS student in class 10th");
   }
   printf("Name = %s
", name );
   printf("Description: %s
", description );
}
```

Explore our **latest online courses** and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

Output

```
Name = Zara Ali
Description: Zara ali a DPS student in class 10th
```

Same program can be written using calloc(); only thing is you need to replace malloc with calloc as follows −

```
calloc(200, sizeof(char));
```

So you have complete control and you can pass any size value while allocating memory, unlike arrays where once the size defined, you cannot change it.

Resizing Memory Locations

When your program comes out, operating system automatically release all the memory allocated by your program but as a good practice when you are not in need of memory anymore then you should release that memory by calling the function free().

Alternatively, you can increase or decrease the size of an allocated memory block by calling the function realloc(). Let us check the above program once again and make use of realloc() and free() functions –

Example Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
   char name[100];
   char *description;
   strcpy(name, "Adam");
   /* allocate memory dynamically */
   description = malloc( 30 * sizeof(char) );
   if( description == NULL ) {
      fprintf(stderr, "Error - unable to allocate required memory
");
   } else {
      strcpy( description, "Adam a DPS student.");
   }
   /* suppose you want to store bigger description */
   description = realloc( description, 100 * sizeof(char) );
   if( description == NULL ) {
      fprintf(stderr, "Error - unable to allocate required memory
");
   } else {
      strcat( description, "He is in class 10th");
   }
   printf("Name = %s
", name );
   printf("Description: %s
", description );
   /* release memory using free() function */
   free(description);
}
```

Output

```
Name = Adam
Description: Adam a DPS student.He is in class 10th
```

You can try the above example without re-allocating extra memory, and strcat() function will give an error due to lack of available memory in description.