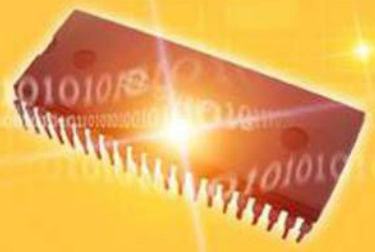
凌阳教育
www.sunplusedu.com
值得信赖的教育品牌

嵌入式系统工程师



Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

动态内存分配

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

动态内存分配

➢ 动态内存分配

- 在数组一章中，介绍过数组的长度是预先定义好的，在整个程序中**固定不变**;
- 但是在实际的编程中，往往会发生这种情况，即所需的**内存空间取决于实际输入的数据**，而无法预先确定
- 为了解决上述问题，C语言提供了一些**内存管理函数**，这些内存管理函数可以按需要**动态的分配**内存空间，也可把不再使用的空间回收再次利用.

动态内存分配

- **静态分配**
  - 在程序编译或运行过程中，按事先规定大小分配内存空间的分配方式.
  - 必须事先知道所需空间的大小.
  - 分配在栈区或全局变量区，一般以数组的形式
  - 按计划分配.
- **动态分配**
  - 在程序运行过程中，根据需要大小自由分配所需空间.
  - 按需分配.
  - 分配在堆区，一般使用特定的函数进行分配.

动态内存分配

1）分配内存空间函数 malloc

函数原型： void *malloc(unsigned int num_bytes);

调用形式： (类型说明符*) malloc (size)

功能说明:

在内存的动态存储区(堆区)中分配一块长度为size字节的连续区域，用来存放类型说明符指定的类型。

函数原型返回void*指针，使用时必须做相应的强制类型转换

分配的内存空间内容不确定，一般使用memset初始化。

返回值: 分配空间的起始地址 （ 分配成功 ）

NULL （ 分配失败 ）

动态内存分配

➢ 例:

01.malloc.c

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4  int main()
5  {
6      int count,*array,n;
7      printf("请输入要申请数组的个数:\n");
8      scanf("%d",&n);
9      array=(int *) malloc ( n * sizeof (int) );
10     if(array==NULL)
11     {
12         printf("申请空间失败!\n");
13         return -1;
14     }
15     //将申请到空间清0
16     memset(array,0,sizeof(int)*n);
17
18     for (count = 0;count < n;count++) /*给数组赋值*/
19         array[count]=count;
20
21     for(count = 0;count < n;count++) /*打印数组元素*/
22         printf("%2d",array[count]);
23     free(array);
24     return 0;
25 }
```

动态内存分配

2）分配内存空间函数 calloc

调用形式: (类型说明符*) calloc (n,size)

功能说明: 在内存动态存储区中分配n块长度为size字节的连续区域.

返回值: 为该区域的首地址.

- calloc与malloc的区别:
  - calloc能一次分配**n块连续**区域
  - calloc能对分配的空间初始化为0，而malloc不能

动态内存分配

- 例:
02.calloc.c
对比malloc

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  int main()
4  {
5      int count,*array,n;
6      printf("请输入要申请数组的个数:\n");
7      scanf("%d",&n);
8      array=(int *) calloc ( n , sizeof (int) )
9      if(array==NULL)
10     {
11         printf("申请空间失败!\n");
12         return -1;
13     }
14
15     for (count = 0;count < n;count++) /*给数组赋值*/
16         array[count]=count;
17
18     for(count = 0;count < n;count++) /*打印数组元素*/
19         printf("%2d",array[count]);
20     free(array);
21     return 0;
22 }
```

动态内存释放

3）realloc

原型:

```
void *realloc(void *mem_address, unsigned int newsize);
```

调用形式:

指针名=（数据类型*）realloc（要改变大小的指针，新的大小）

功能说明:

判断当前的指针后面是否有足够的连续空间，如果有，扩大mem_address指向的地址并且将mem_address返回

若空间不够，先按照newsize指定的大小分配空间，将原有数据拷贝到新分配的内存区域，而后释放原来mem_address所指内存区域

返回新分配的内存区域的首地址。

动态内存分配

➢ 例:
03.realloc.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  int main()
4  {
5      int i;
6
7      int *pn=(int *)malloc(5*sizeof(int));
8
9      for(i=0;i<5;i++)
10         scanf("%d",&pn[i]);
11
12     pn=(int *)realloc(pn,10*sizeof(int));
13
14     for(i=5;i<10;i++)
15         scanf("%d",&pn[i]);
16
17     for(i=0;i<10;i++)
18         printf("%d ",pn[i]);
19
20     free(pn);
21     return 0;
22 }
```

动态内存释放

4）释放内存空间函数free

调用形式： free(void* ptr);

功能说明：

释放ptr所指向的一块内存空间，ptr是一个任意类型的指针变量，指向被释放区域的首地址

注意：

① 被释放区应是由malloc、calloc、realloc函数所分配的区域

② 不释放可能造成内存利用率降低

③ 对同一内存空间多次释放会出错

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com