第11章 动态数据结构的C语言实现

内存映像

哈尔滨工业大学
赵玲玲 *zhaoll@hit.edu.cn*

动态内存分配函数

* 两种基本方式

```
void* malloc(unsigned int size);
```

向系统申请大小为size的内存块，
系统找到一块未占用的内存，将其标记为已占用，
然后把首地址返回，若申请不成功则返回NULL

```
void* calloc(unsigned int num, unsigned int size);
```

向系统申请num个size大小的内存块，
系统找到一块未占用的内存，将其标记为已占用，
然后把首地址返回，若申请不成功则返回NULL

动态内存分配函数

```
#include <stdlib.h>
void* malloc(unsigned int size);
void* calloc(unsigned int num,unsigned int size);
```

- 问题1：怎么申请一块可存放10个整型变量的内存？

```
malloc( 40 );
malloc( 10 * sizeof(int) );
```

动态内存分配函数

```
#include <stdlib.h>
void* malloc(unsigned int size);
void* calloc(unsigned int num,unsigned int size);
```

- 问题2：void * 是什么？
- void*型指针不指定其指向哪一种类型，可指向任意类型的变量，是一种generic或typeless类型的指针.
- 使用时，需强转(Type*)为其他类型

```
p =      malloc( n * sizeof(int) );
```

动态内存分配函数

```
#include <stdlib.h>
void* malloc(unsigned int size);
void* calloc(unsigned int num,unsigned int size);
```

- 问题2：void * 是什么？
- void*型指针不指定其指向哪一种类型，可指向任意类型的变量，是一种generic或typeless类型的指针.
- 使用时，需强转(Type*)为其他类型

```
p = (int *)malloc( n * sizeof(int) );
```

动态内存分配函数

```
int *p1 = NULL;
void *p2;
```

* 空指针p1，与void*类型指针p2不同
  * p1 值为NULL的指针，即无效指针
  * p2 可指向任意类型
  * 既然0（NULL）用来表示空指针，那么空指针就是指向地址为0的单元的指针吗？
    * 不一定．每个C编译器都被允许用不同的方式来表示空指针

空指针与无类型的指针

- 空指针的用途
  - 定义指针时进行初始化，避免对未赋值指针的引用
  - 在程序中常作为状态比较

```
p = (int *) malloc(n * sizeof (int));
  if (p == NULL) //判断p是否为空指针
{
            printf("No enough memory!\n");
            exit(0);
}
```

动态内存分配函数

```
#include <stdlib.h>
void* malloc(unsigned int size);
void* calloc(unsigned int num, unsigned int size);
```

- 从安全的角度考虑，使用calloc()更明智，因为与malloc()不同的是calloc()能自动将分配的内存初始化为0

动态内存分配函数----realloc()

* realloc()用于改变原来分配的存储空间的大小：

  void *realloc(void *p, unsigned int size);

  * 将指针*p*所指向的存储空间的大小改为size个字节
  * 函数返回值是新分配的存储空间的首地址
  * 与原来分配的首地址不一定相同

动态内存分配函数

- 释放（deallocating）内存的方法:

```
void free(void* p);
```

  - 释放由malloc()和calloc()申请的内存块
  - p是指向此块内存的指针
  - free时系统将此块内存标记为未占用，可被重新分配