清華大學 Tsinghua University

腾讯会议ID： 402 743 770 / 2021

计算机程序设计基础

第14讲 编译预处理及动态分配内存

沈瑜（010-62782951）

shenyu@tsinghua.edu.cn

清华大学电机系

2021.12.14

主要内容

- 编译预处理
- 命令行处理
- 动态分配内存

参考教材：**8.7.3， 8.8**

问题？

- 大程序怎么编？
  - 多个.cpp、多个.h
  - 程序的命令行参数？
  - 数组，动态大小？

14.1 编译预处理

1. 编译预处理

编译预处理是指在编译系统对文件进行编译——词法分析、语法分析、代码生成及优化之前，对一些特殊的编译语句先进行处理，然后将处理的结果与源程序一起编译，生成目标文件

编译预处理语句都是以#开头，其结尾不带分号（;）

编译预处理语句分为三类：宏定义、文件包含和条件编译等，它常用于程序设计的模块化、移植、调试等方面。

2. 宏定义

- 宏定义分为两种：不带参数的宏定义（即常量定义）与带参数的宏定义

不带参数的宏定义

#define 标识符 表达式

#define PI 3.1415926

带参数的宏定义

带参数的宏定义中，宏替换名可以带有形式参数，在程序中用到时，实际参数会代替这些形式参数

#define 宏名（参数表） 表达式 //max.cpp

```
#define  MAX(a,b)  (a>b?a:b)

int main()
{
    int a=2, b=4, m;
    m=MAX(a,b);
    printf("%d\n",m);
    return 0;
}
```

带参数的宏定义与函数的区别

① 宏定义仅是对字符串作简单替换
　而函数调用则是按程序的含义来替换形式参数；
　实参如果是表达式容易出问题

```
#define S(r) r*r   //这句可能出bug哦
area=S(a+b);
                   //第一步换为area=r*r;
                   //第二步被换为area=a+b*a+b;
```

正确的宏定义是

```
#define S(r) ((r)*(r))
```

②宏替换只作替换，不做计算，不做表达式求解；
③宏名和参数的括号间不能有空格；
④函数调用在编译后程序运行时进行，并且分配内存。宏替换在编译前进行，不分配内存。

……

- 宏的作用范围

  宏的作用范围，默认情况下，从定义点开始，到程序源文件的末尾。

或者，可使用命令**#undef**取消宏定义

3. 文件包含 #include

- 文件包含的意义

  文件包含的意义是源程序中包含另一个源程序文件。

  一个大型的程序通常都是分为多个模块，由不同的程序员编写，最终需要将它们汇集在一起进行编译。

  另外，在程序设计中，有一些程序代码会经常使用，比如程序中的函数、宏定义等。

  为了方便代码的重用和包含不同模块文件的程序，C语言提供了文件包含的方法。

4. 条件编译

条件编译是指在特定的条件下，对满足条件和不满足条件的情况进行分别处理——满足条件时编译某些语句，不满足条件时编译另一些语句。

- **#if，#elif，#else和#endif**

- **#if**用于对程序进行部分编译，用法与选择语句**if**相似。
- **#elif**的作用类似于**else if**，用于产生多重条件编译。
- **#endif**用于结束条件编译，编译时与前面最近的**#if**作为一对，编译两者之间的部分程序段。

- 条件编译有三种形式，如下所示：

（1）#if …#endif

#if 表达式

　　程序段

#endif

（2）#if…#else…

#if 表达式

程序段**1**

#else

程序段**2**

#endif

（3）#if…#elif…

#if 表达式1
程序段1
#elif 表达式2
程序段2
…
#else 表达式n
程序段n
#endif

- **#ifdef和#ifndef**

语句使用形式如下：

（1）#ifdef

```
#ifdef 宏定义标识符
 /* 如果标识符已定义过，则编译以下程序段*/
   程序段
#endif
```

（2）#ifndef

```
#ifndef 宏定义标识符
/* 如果标识符未定义过，则编译以下程序段*/
   程序段
#endif
```

5. 程序移植和调试

- 程序移植

不同操作系统下，字节数不同

```
//MyEcho.cpp
#ifdef  UNIX
/* 如果在UNIX上，编译这部分程序段*/
        …
#else
/* 否则编译这部分程序段*/
        …
#endif
```

程序调试

在程序进行调试时也可以添加条件编译，在调试时通过**printf()**语句显示出程序变量的变化；

而在调试完毕后利用条件编译将**printf()**语句屏蔽。

6.多个c文件联编

- 编写大文件时，使用自定义头文件(一般为*.h)

  头文件的作用(head file)

  - 定义常用的符号常量、函数声明、外部变量，以便在多个c文件中共用；
  - #include <file.h>时，认为该文件是系统标准头文件
  - #include "file.h"时，先到c文件同目录下查找；若找不到则按系统标准头文件查找。一般用"file.h"

- 如何使用外部变量（全局变量）
  - 全局变量在其中任何一个**c**文件中定义；注意不要**include**进其他**c**文件；
  - 在所有***.c**文件中只能定义一次！在其他需要对该全局变量操作的**c**文件中，增加外部变量声明；
  - 外部变量声明建议放入***.h**头文件，**include**进相关**c**文件
- 例子：

DemoH_main.cpp

DemoH_funcs.cpp

DemoH_common.h

名空间

namespace

14.2 命令行处理

- 命令行举例：

  type a.txt

  type /?

- 如何使用命令行参数？→使用带参数的**main**函数

  main(int argc, char *argv[ ])

  – **argc**为参数个数，第**1**个是**exe**名本身。有效参数个数是**argc-1**个！ **argv**是字符串指针的数组；

  – **argv**的第 **1** 个元素，即**argv[0]**，是程序名，也就是你的可执行文件的名字。

14.3 动态分配内存

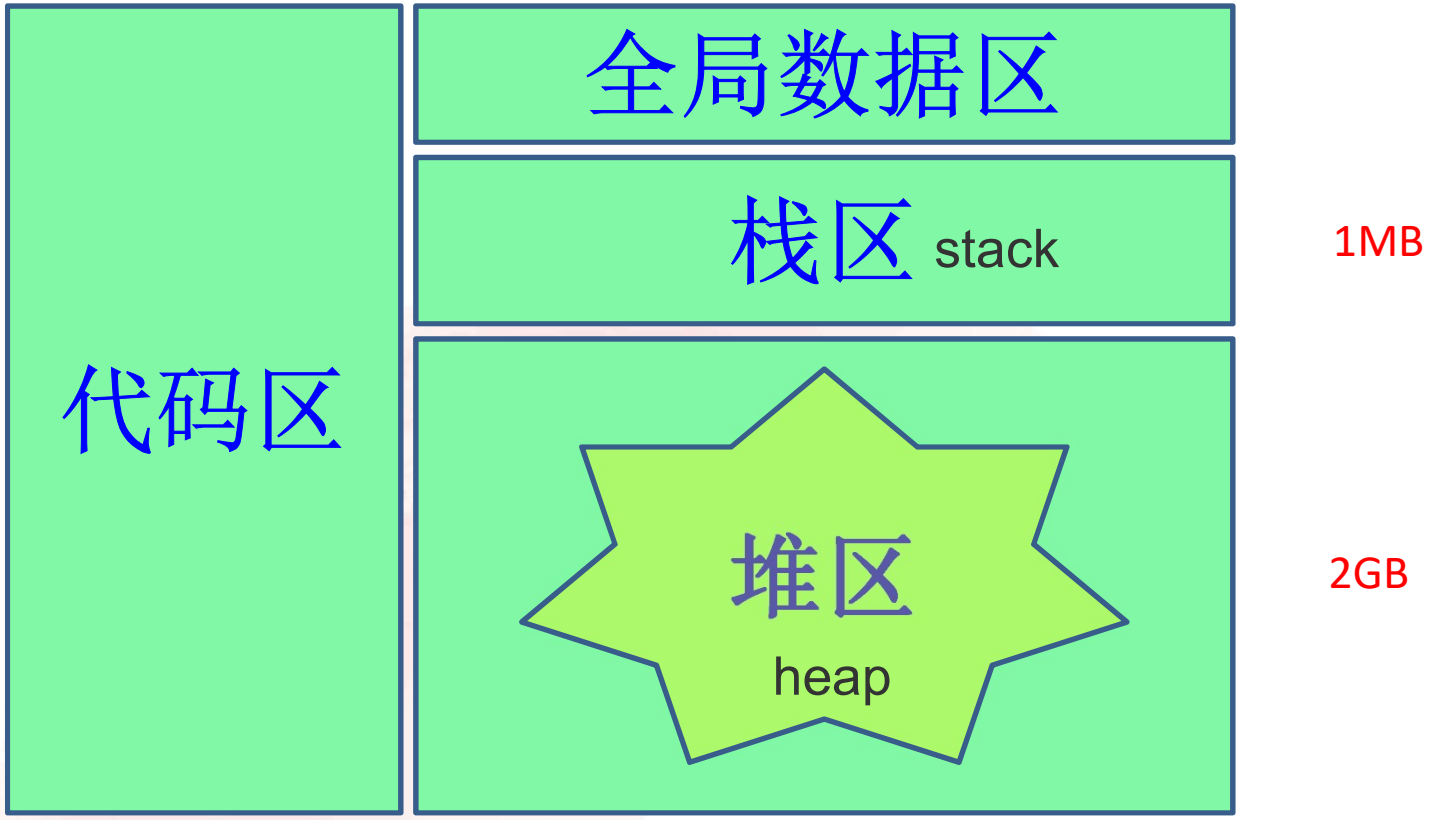
1. 何时需要动态分配内存？

数组大小不清楚

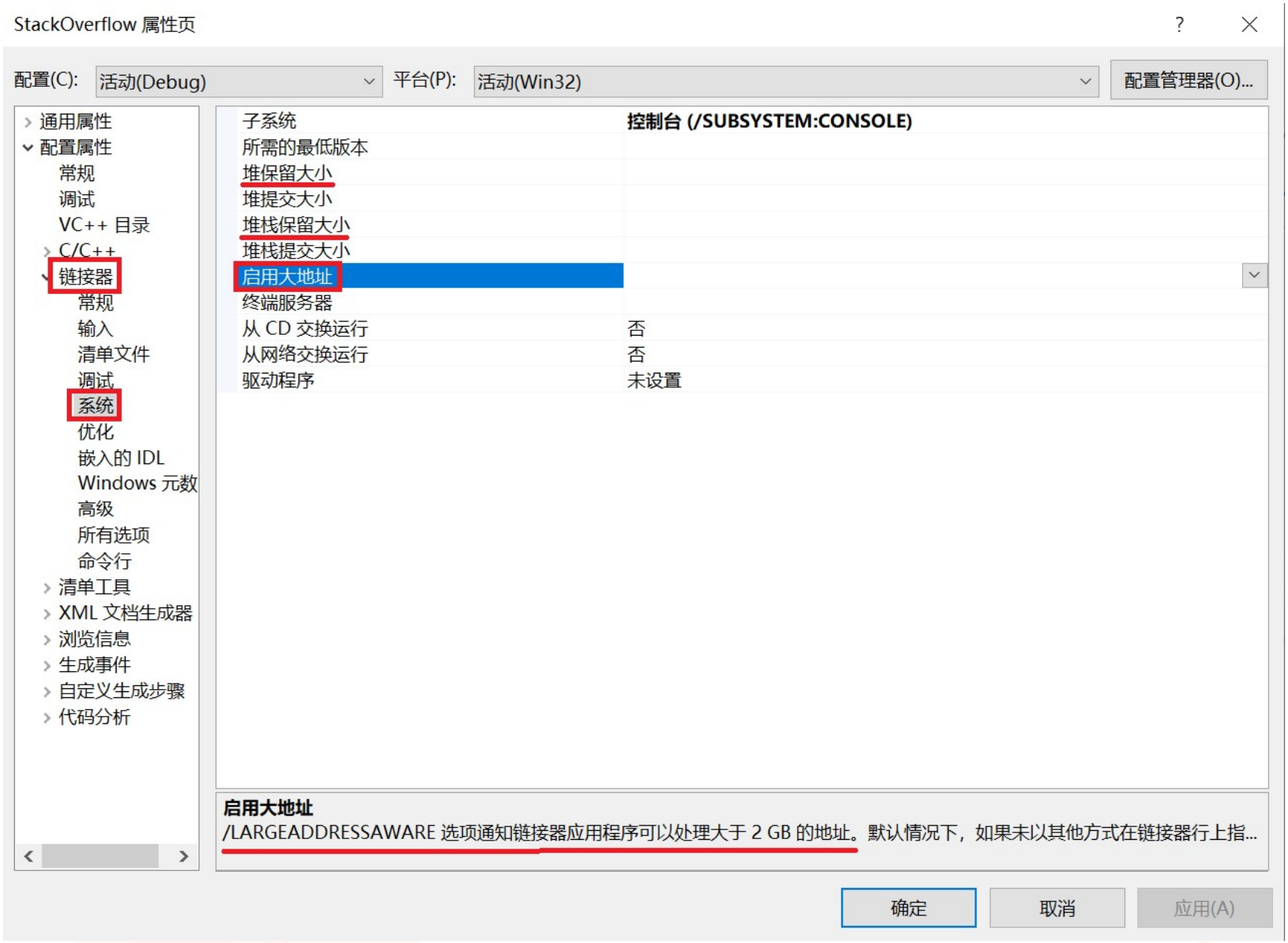
如果数组过大，可能导致**stack overflow**

```
#include <stdio.h>
#define LENGTH  260000
int main()        /*主函数*/
{
    printf("%d", sizeof(int)*LENGTH );
    int naStudent[LENGTH];
    return 0;
}
```

2. 内存的规划



堆区可以分配足够大的空间


StackOverflow 属性页
配置(C): 活动(Debug)
平台(P): 活动(Win32)
配置管理器(O)...
通用属性
配置属性
常规
调试
VC++ 目录
C/C++
链接器
常规
输入
清单文件
调试
系统
优化
嵌入的 IDL
Windows 元数
高级
所有选项
命令行
清单工具
XML 文档生成器
浏览信息
生成事件
自定义生成步骤
代码分析
子系统 控制台 (/SUBSYSTEM:CONSOLE)
所需的最低版本
堆保留大小
堆提交大小
堆栈保留大小
堆栈提交大小
启用大地址
终端服务器
从 CD 交换运行 否
从网络交换运行 否
驱动程序 未设置
启用大地址
/LARGEADDRESSAWARE 选项通知链接器应用程序可以处理大于 2 GB 的地址。默认情况下，如果未以其他方式在链接器行上指...
确定
取消
应用(A)

3. malloc

malloc函数原型：

```
void * malloc(size_t n);
```

n是要分配的内存的大小，返回值是分配内存的块的首地址

```
int * array;
array = (int *)malloc(10 * sizeof(int));
```

注意：内存大小不能写成数组元素的个数

```
struct test
{
    int a;
    char b;
    int c[10];
};

struct test * p;
p = (struct test *)malloc(sizeof(struct test));
```

注意事项：

(1)malloc函数是一个库函数，它并不是C语言中的关键字：

- 需要头文件<stdlib.h>才可以使用该函数
- 并不是所有的平台都可以使用该函数，尤其是一些单片机系统

(2)指针类型转换是必须的，关系到接收分配好的内存块的地址可以向前看多少字节。

(3)内存块大小的可移植性问题

分配一个整型变量数组应使用：

数组元素个数 * sizeof(int)

确定内存块的大小

问题：sizeof和strlen函数的区别

4. free

free函数原型：

void free(void * p);

p是要释放的已分配内存的块的首地址

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

free(p);
```

注意事项：

- 所有分配的内存，程序必须负责释放
- 释放后，请将指针赋为**NULL**
- 不能出现野指针和悬空指针

//pointer.cpp

```
void f()
{
    char*dp;// dp 未初始化，是野指针
}
```

```
int main()
{
    char*dp = NULL;
    for(int i=0;i<1;i++)
    {
        char c;
        dp =&c;
    }// dp 此时为悬空指针
}
```