西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

设计函数 - 函数间的数据传递（1）

3、同一文件的对象冲突与共享

4、不同文件的对象冲突与共享

4.6.3 作用域

- 实体在作用域内可以使用称为可见（visible），又称有效。可见的含义是指实体在作用域上处处可以使用，下面给出C语言实体可见规则。
- （1）规则一。同一个作用域内不允许有相同名字的实体，不同的作用域的实体互不可见，可以有相同名字。
- （2）规则二。实体在包含它的作用域内，从定义或声明的位置开始，按文件行的顺序往后（往下）直到该作用域结束均是可见的，包含作用域内的所有子区域及其嵌套，但往前（往上）不可见，同时包含该作用域的上一级区域也不可见。

4.6.3 作用域

- （3）规则三。若实体A在包含它的作用域内的子区域中出现了相同名字的实体B，则实体A被屏蔽（hide），即实体A在子区域不可见，在子区域中可见的是实体B。
- （4）规则四。可以使用extern声明将变量或函数实体的可见区域往前延伸，称为前置声明（forward declaration）。
- （5）规则五。在全局作用域中，变量或函数实体若使用static修饰，则该实体对于其他源文件是屏蔽的，称为私有的（private）。

4.6.3 作用域

- ①extern声明变量实体的形式为：

```
extern 类型 变量名, ......
```

- extern声明函数原型的形式为：

```
extern 返回类型 函数名(类型1 参数名1,类型2 参数名2,......);
```

```
extern 返回类型 函数名(类型1, 类型2, ......);
```

4.6.3 作用域

- ②static修饰变量实体的形式为：

```
static 类型 变量名[=初值], ......
```

- static修饰函数原型的形式为：

```
static 返回类型 函数名(类型1 参数名1,类型2 参数名2,......);
```

```
static 返回类型 函数名(类型1, 类型2, ......);
```

4.6.3 作用域

- 实体可见规则适用于对象定义和类型声明，前面的局部变量和全局变量也是按这些规则来处理的。

- 下面通过一个程序例子来说明实体可见规则，该程序有两个源文件file1.c和file2.c，file1文件主要说明实体在文件作用域、函数作用域、块作用域的可见规则，file2文件主要说明实体在全局作用域的可见规则，可以参考注释来分析可见规则。

4.6.3 作用域

```
1  // FILE1.C 全局作用域
2  int a=1 , b=2;  //全局变量
3  int c=10 , d=11;  //全局变量
4  void f1(int n,int m)  //f1函数作用域
5  {
6    int x=21, y=22,z=23;  //f1局部变量
7    extern int h,k; //正确，h=60 k=61 规则四
8    n = n + t ; //错误，t 违反规则二
9    if (n>100) {  //块作用域
   ......
20 }
21 int e=50 , f=51;  //全局变量
22 int h=60 , k=61;  //全局变量
```

（1）第7行使用extern声明将第22行的h、k的作用域向前延伸了，故在f1函数中可以使用h、k。

4.6.3 作用域

```
1  // FILE1.C 全局作用域
2  int a=1 , b=2;  //全局变量
3  int c=10 , d=11;  //全局变量
4  void f1(int n,int m)  //f1函数作用域
5  {
6    int x=21, y=22,z=23;  //f1局部变量
7    extern int h,k; //正确, h=60 k=61 规则四
8    n = n + t ; //错误, t 违反规则二
9    if (n>100) {  //块作用域
10     int x=31,t=20;  //复合语句局部变量
11     n = x + y;  //正确, n=31+22 规则二 规则三
12     if (m>10) { //嵌套块作用域
```

（2）根据规则二，不能在函数中使用复合语句的变量，第8行试图使用第10行的变量t是错误的。

4.6.3 作用域

```
 4 void f1(int n,int m)  //f1函数作用域
 5 {
 6   int x=21, y=22,z=23;  //f1局部变量
 7   extern int h,k; //正确，h=60 k=61 规则四
 8   n = n + t ; //错误，t 违反规则二
 9   if (n>100) {  //块作用域
10     int x=31,t=20;  //复合语句局部变量
11     n = x + y;  //正确，n=31+22 规则二 规则三
12     if (m>10) { //嵌套块作用域
13       int y=41; //嵌套的复合语句局部变量
14       n = x + y;  //正确，n=31+41 规则二 规则三
15     }
16   }
```

（3）根据规则三，第14行的x是第10行定义的，第14行的y是第13行定义的。

4.6.3 作用域

```
17    n = a + x ; //正确，n=1+21   规则二
18    m = e + f ; //错误，e,f 违反规则二
19    n = h + k ; //正确，n=60+61 规则四
20 }
21 int e=50 , f=51;  //全局变量
22 int h=60 , k=61;  //全局变量
23 void f2(int n,int m)  //f2函数作用域
24 {
25    n=a+b+e+f;   //正确，n=1+2+50+51 规则二
26    m=z;      //错误，z 违反规则一
27 }
```

（4）因为全局变量e和f在第21行定义，根据规则二作用域是达不到第18行的，除非使用extern声明。

4.6.3 作用域

```
 4 void f1(int n,int m)  //f1函数作用域
 5 {
 6   int x=21, y=22,z=23;  //f1局部变量
 ......
19   n = h + k ; //正确, n=60+61 规则四
20 }
21 int e=50 , f=51;  //全局变量
22 int h=60 , k=61;  //全局变量
23 void f2(int n,int m)  //f2函数作用域
24 {
25   n=a+b+e+f;  //正确, n=1+2+50+51 规则二
26   m=z;     //错误, z 违反规则一
27 }
```

（5）尽管第6行定义了x、y、z，但第26行与第6行是两个不同的函数，不同的作用域是不能相互访问的。

4.6.3 作用域

```
1  // FILE2.C 全局作用域
2  int a=201 , b=202;  //错误，连接时与FILE1.C的同名 违反规则一
3  void f1(int n,int m)  //错误，连接时与FILE1.C的同名 违反规则一
4  {
5    n=n*m;
6  }
7  static int c=210 , d=212; //正确，规则五
8  static void f2(int n,int m) //正确，规则五
9  {
10   n=n/m;
11 }
```

（1）编译器是按文件编译的，所以在编译时未发现错误，但连接时会发现第2行的a、b在全局作用域内已经有定义（FILE1中定义的）；同理，第3行的f1函数也是如此。

4.6.3 作用域

```
1  // FILE2.C 全局作用域
2  int a=201 , b=202;  //错误，连接时与FILE1.C的同名 违反规则一
3  void f1(int n,int m)  //错误，连接时与FILE1.C的同名 违反规则一
4  {
5    n=n*m;
6  }
7  static int c=210 , d=212; //正确，规则五
8  static void f2(int n,int m) //正确，规则五
9  {
10   n=n/m;
11 }
```

（2）第7行又在全局作用域内定义了c、d（FILE1中已定义了），但它有static修饰，根据规则五，FILE1看不到c、d，所以它是正确，第8行同理。

4.6.3 作用域

```
12 extern int h , k; //正确，规则四
13 extern int f4(int n,int m); //正确，规则四
14 int main()
15 {
16   int p,q,r;  //main函数局部变量
17   p = c + d;  //正确，p=210+212 规则二
18   f2(-1,-2);  //正确，不是FILE1.C的 规则二
19   q = e + f;  //错误，试图使用FILE1.C的 违反规则一
20   f3(-10,-12);  //错误，试图使用FILE1.C的 违反规则一
21   r = h + k;  //正确，r=60+61 规则四
22   f4(-20,-22);  //正确，规则四
23   return 0;
24 }
```

（3）根据规则四，第12行将FILE1中的h、k的作用域向前延伸到FILE2文件中来，所以可以在FILE2第21行使用它。

4.6.3 作用域

```
12 extern int h , k; //正确，规则四
13 extern int f4(int n,int m); //正确，规则四
14 int main()
15 {
16   int p,q,r;  //main函数局部变量
17   p = c + d;  //正确，p=210+212 规则二
18   f2(-1,-2);  //正确，不是FILE1.C的 规则二
19   q = e + f;  //错误，试图使用FILE1.C的 违反规则一
20   f3(-10,-12);  //错误，试图使用FILE1.C的 违反规则一
21   r = h + k;  //正确，r=60+61 规则四
22   f4(-20,-22);  //正确，规则四
23   return 0;
24 }
```

（4）第17、18行使用的是FILE2中的定义。

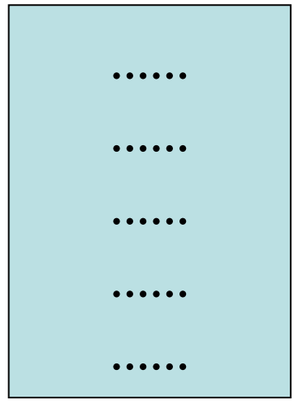
4.6.3 作用域

```
12 extern int h , k; //正确，规则四
13 extern int f4(int n,int m); //正确，规则四
14 int main()
15 {
16   int p,q,r;  //main函数局部变量
17   p = c + d;  //正确，p=210+212 规则二
18   f2(-1,-2);  //正确，不是FILE1.C的 规则二
19   q = e + f;  //错误，试图使用FILE1.C的 违反规则一
20   f3(-10,-12);  //错误，试图使用FILE1.C的 违反规则一
21   r = h + k;  //正确，r=60+61 规则四
22   f4(-20,-22);  //正确，规则四
23   return 0;
24 }
```

（5）根据规则四，第21、22行使用的是FILE1中的定义。

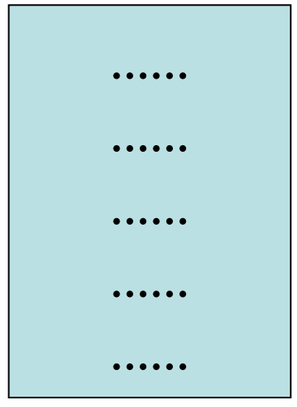
4.6.3 作用域

- 1．同一文件的对象冲突
- 指在一个源程序文件中，有对象名相同引起的冲突。
- 解决办法：
- （1）对象取不同的名称。
- （2）对象设计成局部变量，放入到函数中或者复合语句中，成为“私有的”。



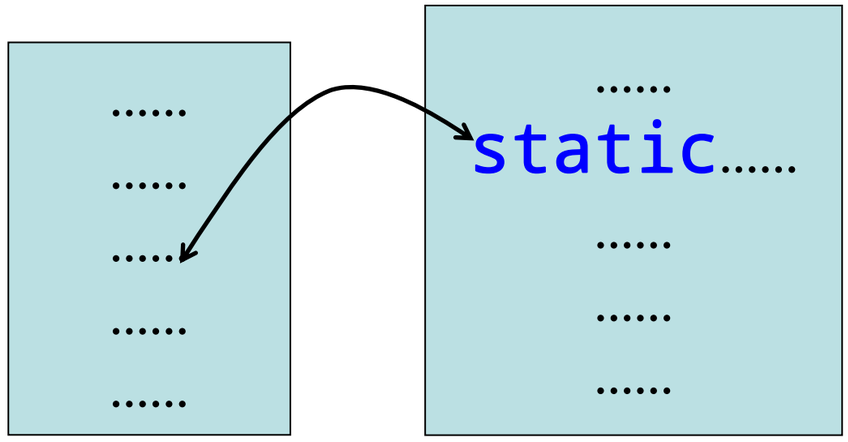
4.6.3 作用域

- 2．同一文件的对象共享
- 指在一个源程序文件中，如何使对象可见。
- 解决办法：
- （1）将对象设计成全局变量。
- （2）用extern把对象声明提前。



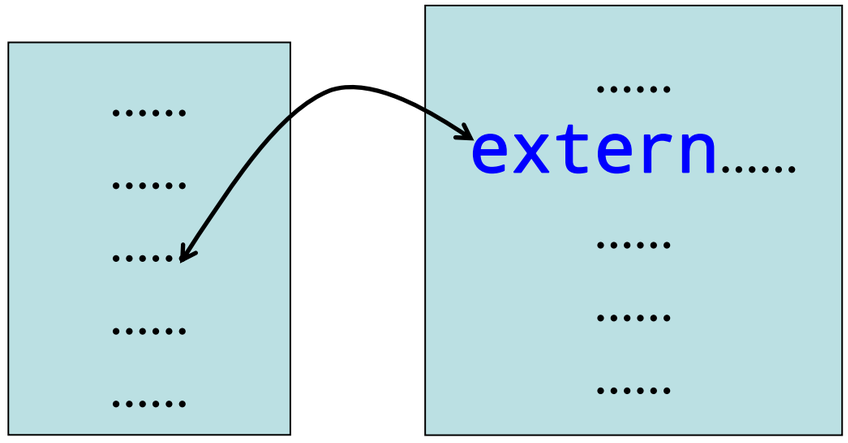
4.6.3 作用域

- 3．不同文件的对象冲突
- 指在2个以上源程序文件中，有对象名相同引起的冲突。
- 解决办法：
- （1）用static将对象设计成文件“私有的”。



4.6.3 作用域

- 4．不同文件的对象共享
- 指在2个以上源程序文件中，如何使对象可见。
- 解决办法：
- （1）用extern把对象声明延伸。




程序设计