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

函数之间传递字符串数据

1、指针参数的const限定

2、字符串作为函数参数

7.5.1 指针作为函数参数

- 3．函数指针变量参数的const限定
- 当函数参数是指针变量时，在函数内部就有可能通过指针间接修改指向对象的值，为避免这个操作可以对指针参数进行const限定，一般形式如下：

```
返回类型 函数名(const 指向类型 *指针变量名,.....)
{
   函数体
}
```

7.5.1 指针作为函数参数

- 例如：

```
void fun1(const char *p,char m)
{
  *p=m; //错误，*p是只读的
  p=&m; //正确，指针变量p可以修改
}
```

- 函数fun不能通过指针变量p修改指向的字符串。
- 如下函数调用：

```
fun1("Hello",c);
```

- 要求第1个形参指针变量的间接引用是只读的，因为实参字符串常量类型是const char *。

7.5.1 指针作为函数参数

- 如果不允许在函数内部修改形参指针变量的值，则定义形式应为：

```
返回类型 函数名(指向类型 * const 指针变量名,.....)
{
   函数体
}
```

- 例如：

```
void fun2(char * const p,char m)
{
  *p=m; //正确 *p是可以修改的
  p=&m; //错误 指针变量p是只读的
}
```

7.5.1 指针作为函数参数

- 4．字符指针变量作为函数形参
- 将一个字符串传递到函数中，传递的是地址，则函数形参既可以用字符数组，又可以用指针变量，两种形式完全等价。在函数中可以修改字符串的内容，主调函数得到的是变化后的字符串。

7.5.1 指针作为函数参数

- 实际编程中，程序员更偏爱用字符指针变量作为函数形参，标准库中很多字符串函数都是这种方式，例如：

```
char *strcpy(char *s1,const char *s2); //字符串复制函数
char *strcat(char *s1,const char *s2); //字符串连接函数
int strcmp(const char *s1, const char *s2); //字符串比较函数
int strlen(const char *s); //计算字符串长度函数
```

7.5.1 指针作为函数参数

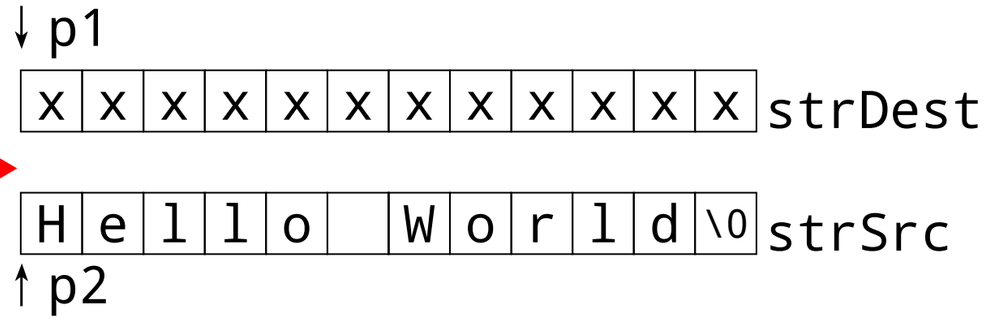
【例7.21】

编写函数stringcpy，实现strcpy函数的字符串复制功能。

7.5.1 指针作为函数参数

例7.21

```
1  #include <stdio.h>
2  char *stringcpy(char *strDest, const char *strSrc)
3  {
4    char *p1=strDest;
5    const char *p2=strSrc;
6    while (*p2!='\0')
7      *p1=*p2, p1++ , p2++;
8    *p1='\0';
9    return strDest; //返回实参指针
10 }
11 int main()
12 {
13   char s1[80],s2[80],s3[80]="string=";
14   gets(s1); //输入字符串
15   stringcpy(s2,s1); //复制s1到s2
```



7.5.1 指针作为函数参数

例7.21

```
16    printf("s2:%s\n",s2);
17    stringcpy(&s3[7],s1); //复制s1到s3的后面
18    printf("s3:%s\n",s3);
19    return 0;
20 }
```

7.5.1 指针作为函数参数

例7.21

程序运行屏幕

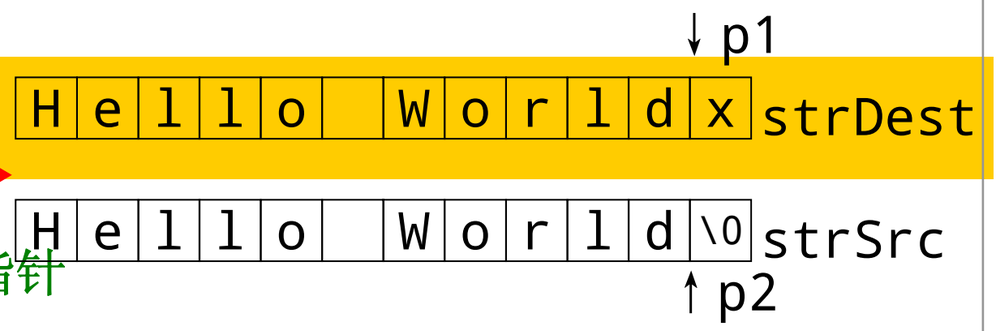
```
s2:Java
s3:string=Java
```

Java↙

7.5.1 指针作为函数参数

例7.21

```
1 #include <stdio.h>
2 char *stringcpy(char *strDest, const char *strSrc)
3 {
4   char *p1=strDest;
5   const char *p2=strSrc;
6   while (*p2!='\0')
7     *p1=*p2, p1++ , p2++;
8   *p1='\0';
9   return strDest; //返回实参指针
10 }
```

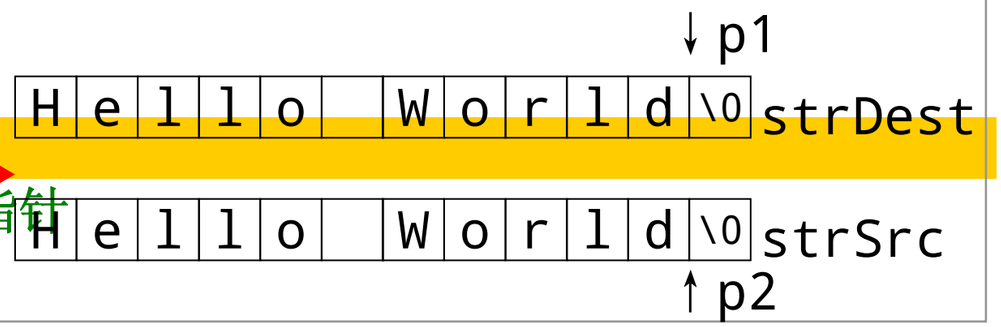


第6、7行是将strSrc字符串每个字符对应地赋值到strDest字符串中，直到结束符'\0'为止。

7.5.1 指针作为函数参数

例7.21

```
1  #include <stdio.h>
2  char *stringcpy(char *strDest, const char *strSrc)
3  {
4    char *p1=strDest;
5    const char *p2=strSrc;
6    while (*p2!='\0')
7      *p1=*p2, p1++ , p2++;
8    *p1='\0';
9    return strDest; //返回实参指针
10 }
```



第8行的作用是在字符复制完成后，在末尾添加一个结束符，使strDest成为字符串。

7.5.1 指针作为函数参数

例7.21

```
1 #include <stdio.h>
2 char *stringcpy(char *strDest, const char *strSrc)
3 {
4   char *p1=strDest;
5   const char *p2=strSrc;
6   while (*p2!='\0')
7     *p1=*p2, p1++ , p2++;
8   *p1='\0';
9   return strDest; //返回实参指针
10 }
```

新字符串生成通常都要有在其末尾添加结束符的操作，明确字符串在合适的位置结束。

7.5.1 指针作为函数参数

```
11 int main()
12 {
13   char s1[80],s2[80],s3[80]="string=";
14   gets(s1); //输入字符串
15   stringcpy(s2,s1); //复制s1到s2
16   printf("s2:%s\n",s2);
17   stringcpy(&s3[7],s1); //复制s1到s3的后面
18   printf("s3:%s\n",s3);
19   return 0;
20 }
```

7.5.1 指针作为函数参数

例7.21

```
1  #include <stdio.h>
2  char *stringcpy(char *strDest, const char *strSrc)
3  {
4    char *p1=strDest;
5    const char *p2=strSrc;
6    while (*p2!='\0')
7      *p1=*p2, p1++ , p2++;
8    *p1='\0';
9    return strDest; //返回实参指针
10 }
```

stringcpy函数定义了两个指针p1和p2来做指针运算，避免形参指针被改变，这样就能够按函数要求返回原始的strDest。

7.5.1 指针作为函数参数

- stringcpy函数第6～8行可以写成更简洁的形式

```
while (*p1=*p2) p1++ , p2++;
```

- 一般地，数值0、空字符'\0'及空指针NULL可以直接当作逻辑值“假”。

7.5.1 指针作为函数参数

例7.21

```
 1 #include <stdio.h>
 2 char *stringcpy(char *strDest, const char *strSrc)
 3 {
 4   char *p1=strDest;
 5   const char *p2=strSrc;
 6   while (*p1=*p2) p1++ , p2++;
 7   return strDest; //返回实参指针
 8 }
 9 int main()
10 {
11   char s1[80],s2[80],s3[80]="string=";
12   gets(s1); //输入字符串
13   stringcpy(s2,s1); //复制s1到s2
14   printf("s2:%s\n",s2);
15   stringcpy(&s3[7],s1); //复制s1到s3的后面
```

7.5.1 指针作为函数参数

例7.21

```
16    printf("s3:%s\n",s3);
17    return 0;
18 }
```

7.5.1 指针作为函数参数

- 5. 指向数组的指针变量作为函数形参
- 函数形参可以是指向数组的指针变量，例如：

```
void swaprow(int (*p1)[4],int (*p2)[4])
{ //交换p1和p2指向的一维数组的元素
  int i,t;
  for (i=0; i<4; i++)
    t=*(*p1+i), *(*p1+i)=*(*p2+i), *(*p2+i)=t;
}
```

7.5.1 指针作为函数参数

- 函数调用时实参也必须是指向数组的指针，例如：

```
int A[4][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
int i=0, j=3;
while (i<j) {
   swaprow(A+i,A+j); //交换A+i和A+j行的元素
   i++, j--;
}
```

7.5.1 指针作为函数参数

- 6. 指向指针的指针变量作为函数形参
- 假设主调函数中有定义

```
int a=10, b=20, *p1=&a, *p2=&b;
```

- 如果一个函数fun的功能是将两个指针的值交换，即函数调用后p1指向b，p2指向a，那么如何设计该函数？

7.5.1 指针作为函数参数

- 根据前面的结论，若要在函数fun中修改p1和p2的值，函数调用就必须用p1和p2的地址作为实参，即

```
fun(&p1,&p2);
```

- 则函数fun不能是

```
void fun(int *x,int *y)
```

- 因为&p1与int *类型不同

7.5.1 指针作为函数参数

- &p1的类型应是指向指针的指针，所以函数fun应如下定义

```
void fun(int **x,int **y) //指向指针的指针变量作为函数形参
{
  int *t; //指针类型
  t=*x, *x=*y, *y=t; //*x和*y为指针类型，两个指针交换
}
```


程序设计