# 12一级指针(char \*)易错模型分析

## 01char \*（字符串）做函数参数出错模型分析

|  |
| --- |
| 建立一个思想：是主调函数分配内存，还是被调用函数分配内存； |
| //不要相信，主调函数给你传的内存空间，你可以写。。。。。。一级指针你懂了。  但是二级指针，你就不一定懂。。。抛出。。。。。。。。。  void copy\_str21(char \*from, char \*to)  {  if (\*from != '\0')  {  printf("ddddd");  }  for (; \*from!='\0'; from++, to++)  {  \*to = \*from;  }  \*to = '\0';  }  //字符串逆序  int main()  {  char \*p = NULL;  char to[100];  copy\_str21(p, to); |
|  |
| 不断修改内存指针变量 |

## 02越界

|  |
| --- |
| 越界 语法级别的越界 |
| char buf[3] = "abc"; |
|  |

## 03不断修改指针变量的值

|  |
| --- |
| 越界 |
|  |
| void copy\_str\_err(char \*from, char \*to)  {  for (; \*from!='\0'; from++, to++)  {  \*to = \*from;  }  \*to = '\0';  printf("to:%s", to);  printf("from:%s", from);  } |
|  |

4、你向外面传递什么

|  |
| --- |
| 1、临时str3内存空间 |
| // char \*str\_cnct(x,y) /\*简化算法\*/  // char \*x,\*y;  char \*str\_cnct(char \*x, char\* y) /\*简化算法\*/  {  char str3[80];  char \*z=str3; /\*指针z指向数组str3\*/  while(\*z++=\*x++);  z--; /\*去掉串尾结束标志\*/  while(\*z++=\*y++);  z=str3; /\*将str3地址赋给指针变量z\*/  return(z);  }  2、经验要学习  while(\*z++=\*x++);  z--; /\*去掉串尾结束标志\*/ |
| char \*str\_cnct(char \*x, char\* y) /\*简化算法\*/  {  char \* str3= (char \*)malloc(80)  char \*z=str3; /\*指针z指向数组str3\*/  while(\*z++=\*x++);  z--; /\*去掉串尾结束标志\*/  while(\*z++=\*y++);  z=str3; /\*将str3地址赋给指针变量z\*/  return(z);  } |
| char \*str\_cnct(char \*x, char\* y) /\*简化算法\*/  {  If (x == NULL)  {  Return NULL;  }  char \* str3= (char \*)malloc(80)  char \*z=str3; /\*指针z指向数组str3\*/  while(\*z++=\*x++);  z--; /\*去掉串尾结束标志\*/  while(\*z++=\*y++);  z=str3; /\*将str3地址赋给指针变量z\*/ note:  return(z);  }  Main ()  {  Char \*p = str\_cnct(“abcd”, “ddeee”);  If (p != NULL) {Free(p) ;p = NULL}//yezhizhen  } |
| int getKeyByValude(char \*keyvaluebuf, char \*keybuf, char \*valuebuf, int \* valuebuflen)  {  int result = 0;  char \*getbuf = new char[100];  memset(getbuf, 0, sizeof(getbuf));  char \*trimbuf = new char[100];  memset(trimbuf, 0, sizeof(trimbuf));  int destlen = strlen(keyvaluebuf);  if (keybuf == NULL || keyvaluebuf == NULL || valuebuf == NULL/\* || valuebuflen == NULL\*/)  {  result = -1;  return result;  }    if (strstr(keyvaluebuf, keybuf) == NULL)  {  result = -1;  return result;  }  else  {  for (int i = 0; i < destlen; i++)  {  if (\*keyvaluebuf == '=')  {  \*keyvaluebuf++;  break;  }  keyvaluebuf++;  }  while(\*keyvaluebuf != '\0')  {  \*valuebuf = \*keyvaluebuf;  valuebuf++;  keyvaluebuf++;  }  \*valuebuf = '\0';  }  int len = strlen(valuebuf);  return result;  } |

|  |
| --- |
| //char \*p = "abcd11111abcd2222abcdqqqqq"; //字符串中"abcd"出现的次数。  //要求你 自己写一个函数接口，并且写出测试用例。  //完成功能为：求出“abcd”字串出现的次数  //输入：  int getSubCount(char \*str, char \*substr, int \* mycount)  {  int ret = 0;  char \*p = str;  char \*sub = substr;  int count = 0;  if (str==NULL || substr==NULL || mycount == NULL)  {  ret = -1;  return ret;  }  //char \*p = "abcd11111abcd2222abcdqqqqqabcd";  //char \*p2 = NULL;  //p2 = p;  do  {  p = strstr(p, sub);  if (p!= NULL)  {  count++;  //++后缀操作符优先级高，所以先执行\*p操作 然后地址++  //(\*mycount)++;  p = p + strlen(sub);  }  else  {  break;  }  } while (\*p != '\0');  //printf("count:%d \n", count);  //mycount是实参的地址 \*（实参的地址）  \*mycount = count;  return ret;  } |
|  |