



Welcome



C标准库

北京亚嵌教育研究中心 ©2011 AKAE







本次课程内容大纲

- ■字符串处理库函数
- ■内存分配库函数
- ■数值字符串转换函数





Section 1

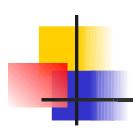


C标准库

字符串处理库函数





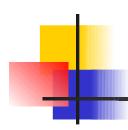


字符串处理库函数

- ■字符串、内存块拷贝库函数
- ■字符串连接库函数
- ■字符串、内存块比较库函数
- ■字符(串)查找库函数
- ■字符串分割库函数



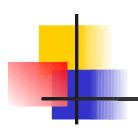




- 基本功能:
 - 将某一块内存区域的数据复制一份到新内存区域中,复制结束条件:
 - ■对于字符串而言:已经复制完了整个字符串(碰到'/0')
 - 对于某一个内存区域而言: 已经复制完了指定字 节数的数据







范例:

将内存区 0xbf3c1000-0xbf3c1002 中存放的字 符串"ab",复制一份到 0xbf3c2000 开始的区域 中。

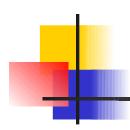
需要知道待复制源串的 起始位置,要复制的字 符个数(长度),以及 存放目标串的起始位置

0xbf3c1000
0xbf3c1001
0xbf3c1002
0xbf3c2000
UNDIJCZUUU
0xbf3c2001

'a'
'b'
'\0'
'a'
'b'
'\0'







- ■据以上分析,字符串、内存块拷贝库函数函数接口如下:
- char *strcpy(char *dest, const char *src)
- void *memcpy(void *dest, const void *src, unsigned n)







- 1、字符串拷贝库函数(复制源字符串内容到一块新的区域,复制到'\0'结束)
 - char *strcpy(char *dest, const char *src)
 - dest 指向的空间足够大(写入)
 - dest 与 src 指向的空间不重叠
 - 以上两点也说明了 strcpy 函数的潜在危险:
 - 如果 dest 空间不够大、 dest 与 src 重叠则 可能会得不到正确答案。







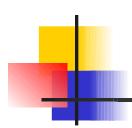
2、字符串拷贝库函数(复制源字符串中最多 n 个字符到一块新的区域,新区域内容不一定以'\0'结束)

char *strncpy(char *dest, const char *src, unsigned n)

- 通常第三个参数 n 用来表示 dest 指向空间的可用字节数,以保证访问不会越界
- dest 与 src 指向的空间不重叠
- 如果 n 小于等于 src 的长度,则目标位置上的内容不会形成家企会法的家籍。







- 3、将某一内存块中的内容复制到新区域中 void *memcpy(void *dest, const void *src, unsigend n)
 - 将从 src 指向的内存区域中的内容复制 n 个 字节到 dest 指向的内存区域中
 - dest 与 src 指向的空间不重叠



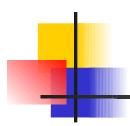




- 4、将某一内存块中的内容复制到新区域中 void *memmove(void *dest, const void *src, unsigend n)
 - 将从 src 指向的内存区域中的内容复制 n 个字 节到 dest 指向的内存区域中
 - dest 与 src 指向的空间可以重叠







- 总结:
- str 开头的库函数与 mem 开头的库函数比较
 - str: 处理以'\0'结尾的字符串
 - mem: 只处理若干字节数据(一块内存区域), 而不关心处理的内容是否是字符串



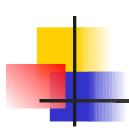




- 函数使用:
 - strcpy:
 - ■char *str, *p = "hello";// 将 p 指向的字符 串复制到 str 数组中
 - ■strcpy(str, p); // 正确吗?







char str[20], *p = "hello";// 将 p 指向的字符串复制到 str 数组中

str = p; // 正确吗?

- ■字符数组不能整体赋值,数组名不能 放在赋值符左侧
- 若想实现字符串整体复制,需要使用 strcpy 函数: strcpy(str, p);







■ 函数使用:

- \blacksquare int num[5] = {1, 2, 3, 4, 5};
- ■int dest[5]; // 将 num 数组中的内容复制到 dest 数组中
- ■dest = num; //正确吗?







■ 函数使用:

■方法一: 一维数组使用一重循环遍历数组,

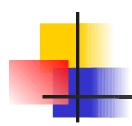
每一次循环完成一个 int 型数据的复制

for(
$$i = 0$$
; $i < 5$; $i++$)
dest[i] = num[i];

- ■方法二:
- memcpy(dest, num, 5 * sizeof(int))
 memmove(dest, num, 20);







■补充:

向某一块内存区域填充指定数值: memset void *memset(void *s, int c, unsigned n)

将从 s 地址开始的 n 个字节填充为数据 c 返回值与 s 相同

通常用于将某块区域清零



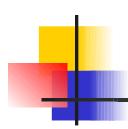




- 练习:
 - 查看书上的 memcpy 函数
 - 查看书上的 memmove 函数







字符串连接库函数

- 在字符串原有内容的基础上再添加新内 容
- strcat、 strncat: 字符串拷贝连接
 •char *strcat(char *s1, char *s2)
 •char *strncat(char *s1, char *s2, unsigned n)
 - •strncat 也是对 strcat 的改进,最后一个参数代表从 s2 指向的字符串中最多取 n 个字符连接到 dest 的末尾,以保证不发生越界







字符串、内存块比较库函数

- 比较字符串或内存块中内容(数值),得出两个字符串或内存区域数据小于、等于、大于的结论。
 - ■比较字符串:按照 ASCII 值进行比较
 - ■比较内存块:按照实际数值进行比较
- 如: strcmp("ab", "abc"): 结论为负值, 表示" ab" 小于" abc"







字符串、内存块比较库函数

- #include<string.h>:区分大小写
- int strcmp(const char *s1, const char *s2):
- int strncmp(const char *s1, const char *s2, unsigend n)
- #include<strings.h>: 不区分大小写
- int strcasecmp(const char *s1, const char *s2):
- int strncasecmp(const char *s1, const char *s2, unsigend n)







字符串、内存块比较库函数

- ■比较内存块
- #include<string.h>
- int memcmp(const void *s1, const void *s2, unsigend n)
- 比较 s1 及 s2 指向区域中的 n 个字节







- ■搜索字符
- 查找一个字符串中是否有指定字符
 - 从左向右查找: strchr
 - 从右向左查找: strrchr







- 搜索字符
- 查找一个字符串中是否有指定字符
 - char * strchr(const char *s, int c)
 - char *strrchr(const char *s, int c)
 - ■若未找到返回值为 NULL (空指针)
 - ■若找到则返回该字符在字符串中所在处的地址 值







■ 搜索字符 strchr("hello", 'l')

返回值

为:0x8048002

strrchr("hello", 'l')

返回值

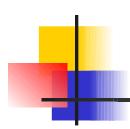
为:0x8048003

0x8048000
0x8048001
0x8048002
0x8048003
0x8048004
0x8048005

'h'
'e'
'1'
'1'
'o'
' \0'







- ■搜索子串
- 查找一个字符串中是否有指定的子字符串
 - char * strstr(const char *src, const char *substr)
 - ■若未找到则返回 NULL
 - ■若找到则返回该字串在源串中的首字符地址值







■ 搜索子串 strstr("abcdefg", "cd")

返回值为 0x8048002

0x8048000
0x8048001
0x8048002
0x8048003
0x8048004
0x8048005
0x8048006
0x8048007

'a'
ʻb'
'c'
'd'
'e'
'f'
ʻg'
' \0'







■ 练习:

统计子串 substr 在源串 src 中出现的次数。

源串及子串数据从命令行参数获取。







■ 练习:

统计子串 substr 在源串 src 中出现的次数。

源串及子串数据从命令行参数获取。







■ 练习:

定义一个字符指针数组如:

从键盘输入字符串,从字符指针数组对应的字符串中查找输入的字符串是否存在,若存在返回该字符串存在于指针数组的行列位置。若输入"exit"(不区分大小写)结束字符串输入







检索字符串

■ int strspn(const char *, const char *); strspn 函数范例 strspn("bbbc", "bcd") ,搜索字符 串" abc" 中是否有不存在于字符集" bcd" 中出现的 字符。如' a' 没有出现在" bcd" 中,则函数返回字 符串" abc" 中' a' 的下标: 0,如果第一个参数代 表的字符串中的所有字符都出现在第二个参数代 表的字符串中则函数返回 '\0' 字符的下标: 3。







检索字符串

■ int strcspn(const char *, const char *); strcspn 函数范例 strcspn("aaaaaa", "bcd"), 搜索字符串"abc"中出现在"bcd"字符集中 的字符返回其下标,则函数应该返 回"abc"中'b'的下标: 1;若第一个参数 中所有字符都没有在第二个参数中出现则返 回'\0'的下标。







分割字符串

- 按照指定的字符把源串分割为更小的字符串
 - char * strtok(char *src, char *delim)
 - ■src 指向待分割源串(该区域可写入)
 - ■delim 分割依据
 - ■在 src 中查找是否有 delim 指向的字符串中的内容,如有则将 src 中出现该字符的位置设置为'\0',返回值为分割出的字串的首地址,分割完毕返回 NULL







字符串分割库函数

■分割字符串

char str[] = "ab:c;d";

第一次调用 strtok(str,

":;")

返回值为

0xbf3c8000:"ab"

第二次调用 strtok(NULL,

"…")

返回值为 0xbf3c8003:"c"

0xbf3c8000
0xbf3c8001
0xbf3c8002
0xbf3c8003
0xbf3c8004
0xbf3c8005
0xbf3c8006

'a'
ʻb'
':' ' \0'
'c'
';' '\0'
'd'
' \0'







字符串分割库函数

■分割字符串

char *strtok_r(char *str, char *delim, char
 **savep)

■ 比 strtok 多一个参数,用于保存每次调用完函数后下一次进行字符串分割时的起始地址,该参数的值是在 strtok_r 中设置的







分割字符串

char str[] = "ab:c;d";

char *pos;

第一次调用 strtok_r(str,

":;", &p)

返回值为 0xbf3c8000:"ab"

第二次调用 strtok_r(NULL

":;", &p)

	0xbf3c8000
•	0xbf3c8001
	0xbf3c8002
	0xbf3c8003
	0xbf3c8004
	0xbf3c8005
	0xbf3c8006
- 4	

'a'
ʻb'
':' ' \0'
'c'
';' '\0'
'd'
' \0'







字符串处理函数

- 注意:
- 使用 strtok、 strtok_r 函数分割函数时 ,待分割的字符串应该保存在一个可写区 域内。







字符串搜索库函数

- ■练习
- 若有字符串 http://www.google.cn/search? complete=1&hl=zh-CN&ie=GB2312&q=linux&meta=
- 用于进行数据搜索,使用库函数分别解析 出 key 与 value 数据







字符串处理函数

- ■练习
- 编写库函数: strtok





Section 2



C标准库

内存分配库函数







- 堆空间分配函数
- void *malloc(size_t size)
- void *calloc(size_t nmemb, size_t size)
- void *realloc(void *ptr, size t size)

■ 以上函数应该与 free 函数成对出现







- 堆空间分配函数
- void *calloc(size_t nmemb, size_t size)
- calloc 函数会将申请的 nmemb*size 个字 节清零







- 堆空间分配函数
- void *realloc(void *ptr, size_t size)
- realloc 函数可在原来申请的堆空间基础上 修改占用的空间大小
- realloc 函数的参数 ptr 可以是 NULL 或malloc、 calloc 函数的返回指针
- realloc(NULL,size) === malloc(size)
- \blacksquare realloc(ptr, 0) ==== free(ptr)







- 课程练习
- 输入若干自然数,若输入数据为 0 则结束输入,把以上数据保存到堆中,然后打印。





Section 3



C标准库

数值字符串转换库函数







- 字符串转为整数
- int atoi(char *s): 不能转换则返回 0
 - 将字符串 s 转为 int 整数
 - atoi("123"):结论为 123
 - atoi("12ab3456"): 结论为 12
 - atoi("ab"): 结论为 0







- 字符串转为整数
- double atof(char *s)
 - 将字符串 s 转为 double 浮点数
 - atof("4.5b"):结论为 4.5
 - atof("4.5e+1c"): 结论为 45
 - atof("ab"): 结论为 0







- 字符串转为浮点数
- double aotf(char *s)
 - 将字符串 s 转为 double 浮点数
 - atof("4.5b"):结论为 4.5
 - atof("4.5e+1c"): 结论为 45
 - atof("ab"): 结论为 0







- 以上两个函数不严谨:
 - atoi("0ab"):结论为 0
 - atoi("ab"): 结论为 0
 - ■这两种情况结论均为 0 ,但是第一个调用为正确值,第二个调用是转换失败时的返回值, atoi 函数对于这两种情况不做区分。







■ 可以使用函数

long strtol(char *nptr, char **endptr, int
 base)

改进 atoi 函数:

nptr: 指向要进行转换的字符串

endptr:函数返回时指向 nptr 中未被识别的第一个字符。

base:表示基数







- char *p;
- strtol("0ab", &p, 10): 结论为 0, p 指向字符' a'
- strtol("cd", &p, 10): 结论为 0, p 指向首字符'c'

■ 没有将 int 型数据转为字符串的库函数,需自己完成函数编写







- 练习:
- 编程实现:

memset, strchr, strstr, strcasecmp,atoi, strtol







■ 附加练习:

编写函数 fun , 其功能是: 找出形参 s 所指字符串中出现频率最高的字母(不区分大小写),并统计出其出现的次数。例如,形参 s 所指的字符串为: abcAbsmaxless,程序执行后的输出结果为:

letter 'a': 3 times letter 's': 3 times







Let's DO it!

Thanks for listening!

