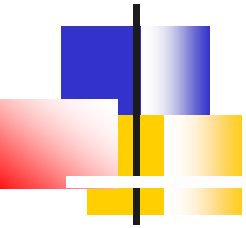
Welcome



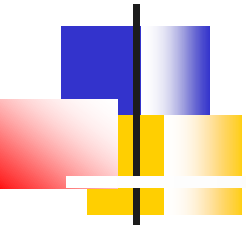
C标准库

北京亚嵌教育研究中心

本次课程内容大纲

- 字符串处理库函数
- 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

字符串、内存块拷贝库函数

- 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

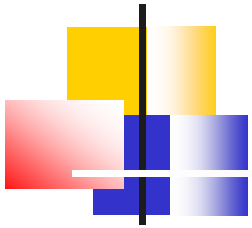
字符串、内存块拷贝库函数

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。

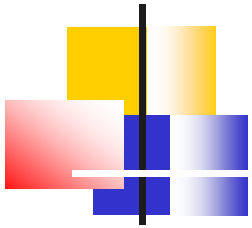
需要知道待复制源串的起始位置，要复制的字符个数（长度），以及存放目标串的起始位置。

0xbf3c1000	‘a’
0xbf3c1001	‘b’
0xbf3c1002	‘\0’
0xbf3c2000	‘a’
0xbf3c2001	‘b’
0xbf3c3002	‘\0’



字符串、内存块拷贝库函数

- 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)
- char *memcpy(void *dest, const void *src, unsigned n)



字符串、内存块拷贝库函数

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)

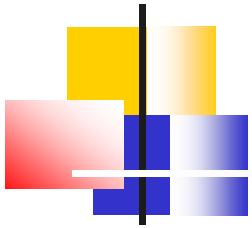
- dest指向的空间足够大(写入）
- dest与src指向的空间不重叠
- 以上两点也说明了strcpy函数的潜在危险：
- 如果dest空间不够大、dest与src重叠则可能会得不到正确答案。

字符串、内存块拷贝库函数

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一个合法的字符串



字符串、内存块拷贝库函数

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

字符串、内存块拷贝库函数

4、将某一内存块中的内容复制到新区域中

void *memmove(void *dest, void *src, unsigend n)

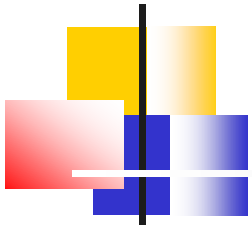
- 将从src指向的内存区域中的内容复制n个字节到dest指向的内存区域中
- dest与src指向的空间可以重叠

字符串、内存块拷贝库函数

- 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

字符串、内存块拷贝库函数

- 函数使用：
  - strcpy：
  - 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？



字符串、内存块拷贝库函数

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)；

字符串、内存块拷贝库函数

- 函数使用：
  - 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？

字符串、内存块拷贝库函数

- 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);
```

字符串、内存块拷贝库函数

■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

字符串、内存块拷贝库函数

- 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

- 比较字符串
- #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

字符串搜索库函数

- 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

字符串搜索库函数

- 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

字符串搜索库函数

- 搜索字符

strchr("hello", 'l')

返回值为:0x8048002

strrchr("hello", 'l')

返回值为:0x8048003

0x8048000	'h'
0x8048001	'e'
0x8048002	'l'
0x8048003	'l'
0x8048004	'o'
0x8048005	'\0'

字符串搜索库函数

- 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

字符串搜索库函数

- 搜索子串

strstr("abcdefg", "cd")

返回值为0x8048002

0x8048000	'a'
0x8048001	'b'
0x8048002	'c'
0x8048003	'd'
0x8048004	'e'
0x8048005	'f'
0x8048006	'g'
0x8048007	'\0'

字符串搜索库函数

- 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。

字符串搜索库函数

- 练习：

定义一个字符指针数组如：

```
char *str[] = {"hello world“, "hello hell“,
               "hello aka“,  "hello hello hoho"};
```

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

字符串搜索库函数

检索字符串

- int strspn(const char *, const char *);

strspn函数范例strspn(“abc”, “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。

字符串搜索库函数

- 检索字符串
- int strcspn(const char *, const char *);

strcspn函数范例strcspn(“abc”, “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。

字符串搜索库函数

- 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	‘a’
0xbf3c8001	‘b’
0xbf3c8002	‘:’ ‘\0’
0xbf3c8003	‘c’
0xbf3c8004	‘;’ ‘\0’
0xbf3c8005	‘d’
0xbf3c8006	‘\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

字符串分割库函数

- 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)

返回值为0xbf3c8003:”c”

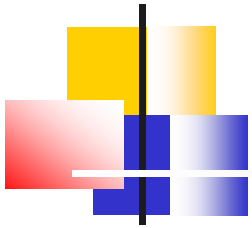
0xbf3c8000	‘a’
0xbf3c8001	‘b’
0xbf3c8002	‘:’ ‘\0’
0xbf3c8003	‘c’
0xbf3c8004	‘;’ ‘\0’
0xbf3c8005	‘d’
0xbf3c8006	‘\0’

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

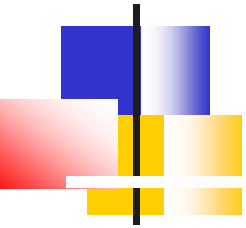
- 练习
- 若有字符串 http://www.google.cn/search?complete=1&hl=zh-CN&ie=GB2312&q=linux&meta=
- 用于进行数据搜索，使用库函数分别解析出key与value数据



字符串处理函数

- 练习
- 编写库函数：strtok

Section 2



C标准库

数值字符串转换库函数

字符串数值转换库函数

- 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(“12ab”):结论为12
  - atoi(“ab”):结论为0

字符串数值转换库函数

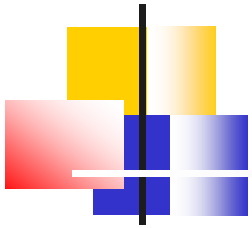
- 字符串转为整数
- double atof(char *s)
  - 将字符串s转为double浮点数
  - atof(“4.5b”) :结论为4.5
  - atof(“4.5e+1c”):结论为45
  - atof(“ab”):结论为0

字符串数值转换库函数

- 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

字符串数值转换库函数

- 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。



字符串数值转换库函数

- 可以使用函数

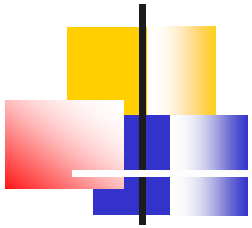
long strtol(char *nptr, char **endptr, int base)

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。

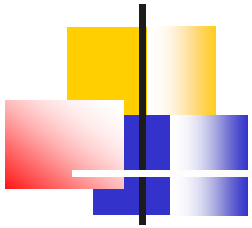
base：表示基数



字符串数值转换库函数

- 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



字符串数值转换库函数

- 练习：
- 编程实现：

memset, strchr, strstr, strcasecmp,atoi, strtol

大作业-命令行解析器

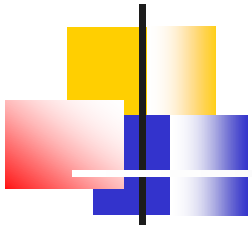
- 任务描述：
  - 使用字符串处理知识实现命令行的解析
  - 要求1：能够解析命令及参数，能够识别管道符

```
akaedu@akaedu:~/120210$ ./shell
sh% ls -l | wc -l
cmd-show:ls,argv:-l,in:(null), out:(null)
cmd-show:wc,argv:-l,in:(null), out:(null)
sh%
```

大作业-命令行解析器

- 任务描述：
  - 使用字符串处理知识实现命令行的解析
  - 要求2：能够解析输入及输出重定向

```
akaedu@akaedu:~/120210$ ./shell
sh%  ls -l > output-file
cmd-show:ls,argv:-l,in:(null), out:output-file
sh% wc -l < input-file
cmd-show:wc,argv:-l,in:input-file, out:(null)
sh%
```



Let's DO it!

Thanks for listening!