C++字符串函数详解[转]2009-07-31 17:29void \*memset(void \*dest, int c, size\_t count);

将dest前面count个字符置为字符c.

返回dest的值.

void \*memmove(void \*dest, const void \*src, size\_t count);

从src复制count字节的字符到dest. 如果src和dest出现重叠, 函数会自动处理.

返回dest的值.

void \*memcpy(void \*dest, const void \*src, size\_t count);

从src复制count字节的字符到dest. 与memmove功能一样, 只是不能处理src和dest出现重叠.

返回dest的值.

void \*memchr(const void \*buf, int c, size\_t count);

在buf前面count字节中查找首次出现字符c的位置. 找到了字符c或者已经搜寻了count个字节, 查找即停止.

操作成功则返回buf中首次出现c的位置指针, 否则返回NULL.

void \*\_memccpy(void \*dest, const void \*src, int c, size\_t count);

从src复制0个或多个字节的字符到dest. 当字符c被复制或者count个字符被复制时, 复制停止.

如果字符c被复制, 函数返回这个字符后面紧挨一个字符位置的指针. 否则返回NULL.

int memcmp(const void \*buf1, const void \*buf2, size\_t count);

比较buf1和buf2前面count个字节大小.

返回值< 0, 表示buf1小于buf2;

返回值为0, 表示buf1等于buf2;

返回值> 0, 表示buf1大于buf2.

int memicmp(const void \*buf1, const void \*buf2, size\_t count);

比较buf1和buf2前面count个字节. 与memcmp不同的是, 它不区分大小写.

返回值同上.

size\_t strlen(const char \*string);

获取字符串长度, 字符串结束符NULL不计算在内.

没有返回值指示操作错误.

char \*strrev(char \*string);

将字符串string中的字符顺序颠倒过来. NULL结束符位置不变.

返回调整后的字符串的指针.

char \*\_strupr(char \*string);

将string中所有小写字母替换成相应的大写字母, 其它字符保持不变.

返回调整后的字符串的指针.

char \*\_strlwr(char \*string);

将string中所有大写字母替换成相应的小写字母, 其它字符保持不变.

返回调整后的字符串的指针.

char \*strchr(const char \*string, int c);

查找字符c在字符串string中首次出现的位置, NULL结束符也包含在查找中.

返回一个指针, 指向字符c在字符串string中首次出现的位置, 如果没有找到, 则返回NULL.

char \*strrchr(const char \*string, int c);

查找字符c在字符串string中最后一次出现的位置, 也就是对string进行反序搜索, 包含NULL结束符.

返回一个指针, 指向字符c在字符串string中最后一次出现的位置, 如果没有找到, 则返回NULL.

char \*strstr(const char \*string, const char \*strSearch);

在字符串string中查找strSearch子串.

返回子串strSearch在string中首次出现位置的指针. 如果没有找到子串strSearch, 则返回NULL. 如果子串strSearch为空串, 函数返回string值.

char \*strdup(const char \*strSource);

函数运行中会自己调用malloc函数为复制strSource字符串分配存储空间, 然后再将strSource复制到分配到的空间中. 注意要及时释放这个分配的空间.

返回一个指针, 指向为复制字符串分配的空间; 如果分配空间失败, 则返回NULL值.

char \*strcat(char \*strDestination, const char \*strSource);

将 源串strSource添加到目标串strDestination后面, 并在得到的新串后面加上NULL结束符. 源串strSource的字符会覆盖目标串strDestination后面的结束符NULL. 在字符串的复制或添加过程中没有溢出检查, 所以要保证目标串空间足够大. 不能处理源串与目标串重叠的情况.

函数返回strDestination值.

char \*strncat(char \*strDestination, const char \*strSource, size\_t count);

将 源串strSource开始的count个字符添加到目标串strDest后. 源串strSource的字符会覆盖目标串strDestination后面的结束符NULL. 如果count大于源串长度, 则会用源串的长度值替换count值. 得到的新串后面会自动加上NULL结束符. 与strcat函数一样, 本函数不能处理源串与目标串重叠的情况.

函数返回strDestination值.

char \*strcpy(char \*strDestination, const char \*strSource);

复制源串strSource到目标串strDestination所指定的位置, 包含NULL结束符. 不能处理源串与目标串重叠的情况.

函数返回strDestination值.

char \*strncpy(char \*strDestination, const char \*strSource, size\_t count);

将 源串strSource开始的count个字符复制到目标串strDestination所指定的位置. 如果count值小于或等于strSource串的长度, 不会自动添加NULL结束符目标串中, 而count大于strSource串的长度时, 则将strSource用NULL结束符填充补齐count个字符, 复制到目标串中. 不能处理源串与目标串重叠的情况.

函数返回strDestination值.

char \*strset(char \*string, int c);

将string串的所有字符设置为字符c, 遇到NULL结束符停止.

函数返回内容调整后的string指针.

char \*strnset(char \*string, int c, size\_t count);

将string串开始count个字符设置为字符c, 如果count值大于string串的长度, 将用string的长度替换count值.

函数返回内容调整后的string指针.

size\_t strspn(const char \*string, const char \*strCharSet);

查找任何一个不包含在strCharSet串中的字符 (字符串结束符NULL除外) 在string串中首次出现的位置序号.

返回一个整数值, 指定在string中全部由characters中的字符组成的子串的长度. 如果string以一个不包含在strCharSet中的字符开头, 函数将返回0值.

size\_t strcspn(const char \*string, const char \*strCharSet);

查找strCharSet串中任何一个字符在string串中首次出现的位置序号, 包含字符串结束符NULL.

返回一个整数值, 指定在string中全部由非characters中的字符组成的子串的长度. 如果string以一个包含在strCharSet中的字符开头, 函数将返回0值.

char \*strspnp(const char \*string, const char \*strCharSet);

查找任何一个不包含在strCharSet串中的字符 (字符串结束符NULL除外) 在string串中首次出现的位置指针.

返回一个指针, 指向非strCharSet中的字符在string中首次出现的位置.

char \*strpbrk(const char \*string, const char \*strCharSet);

查找strCharSet串中任何一个字符在string串中首次出现的位置, 不包含字符串结束符NULL.

返回一个指针, 指向strCharSet中任一字符在string中首次出现的位置. 如果两个字符串参数不含相同字符, 则返回NULL值.

int strcmp(const char \*string1, const char \*string2);

比较字符串string1和string2大小.

返回值< 0, 表示string1小于string2;

返回值为0, 表示string1等于string2;

返回值> 0, 表示string1大于string2.

int stricmp(const char \*string1, const char \*string2);

比较字符串string1和string2大小，和strcmp不同, 比较的是它们的小写字母版本.

返回值与strcmp相同.

int strcmpi(const char \*string1, const char \*string2);

等价于stricmp函数, 只是提供一个向后兼容的版本.

int strncmp(const char \*string1, const char \*string2, size\_t count);

比较字符串string1和string2大小，只比较前面count个字符. 比较过程中, 任何一个字符串的长度小于count, 则count将被较短的字符串的长度取代. 此时如果两串前面的字符都相等, 则较短的串要小.

返回值< 0, 表示string1的子串小于string2的子串;

返回值为0, 表示string1的子串等于string2的子串;

返回值> 0, 表示string1的子串大于string2的子串.

int strnicmp(const char \*string1, const char \*string2, size\_t count);

比较字符串string1和string2大小，只比较前面count个字符. 与strncmp不同的是, 比较的是它们的小写字母版本.

返回值与strncmp相同.

char \*strtok(char \*strToken, const char \*strDelimit);

在strToken 串中查找下一个标记, strDelimit字符集则指定了在当前查找调用中可能遇到的分界符.

返回一个指针, 指向在strToken中找到的下一个标记. 如果找不到标记, 就返回NULL值. 每次调用都会修改strToken内容, 用NULL字符替换遇到的每个分界符.