模拟strlen

size\_t my\_strlen(const char\* str)

{

const char\* begin = str;

assert(str != NULL);

while ((\*str) != '\0')

{

str++;

}

const char\* end = str;

return end-begin;

}

int main()

{

char arr[] = "abcdef";

unsigned int len = my\_strlen(arr);

printf("%u\n", len);

return 0;

}

模拟strcpy

//模拟strcpy

char\* my\_strcpy(char\* dest,const char\* src)

{

assert(dest && src);

char\* begin = dest;

while (\*dest++ = \*src++)

;

\*dest = \*src;

return begin;

}

int main()

{

char des[20] = "";

char src[]="sorakado";

my\_strcpy(des, src);

printf("%s\n", des);

return 0;

}

模拟strcat

char\* my\_strcat(char\* dest,const char\* src)

{

assert(dest && src);

char\* begin = dest;

while (\*dest)

dest++;

while (\*dest++ = \*src++)

;

return begin;

}

实现可自我追加的strcat

char\* my\_strcat(char\* dest, const char\* src)

{

assert(dest && src);

char\* begin = dest;

char\* cpy\_dest = dest;

char\* cpy\_src = (char\*) src;

int count1 = 0;

int count2 = 0;

while (\*dest)

{

dest++;

count1++;

}

while (\*src)

{

src++;

count2++;

}

if (cpy\_dest == cpy\_src && count1==count2)

{

for (int i = 0 ;i < count1; i++)

{

\*dest++ = \*cpy\_src++;

}

\*++dest ='\0';

}

else

while (\*cpy\_dest++ = \*cpy\_src++)

;

return begin;

}

模拟strcmp

int my\_strcmp(const char\* str1,const char\* str2)

{

assert(str1 && str2);

while (\*str1 == \*str2)

{

if (\*str1 == '\0')

return 0;

str1++;

str2++;

}

int ret = \*str1 - \*str2;

return ret;

}

模拟strstr

char\* my\_strstr( char\* str1,const char\* str2)

{

assert(str1 && str2);

char\* begin2 =(char\*) str2;

while (\*str1)

{

if (\*str1 == \*str2) //如果相同，就开始比较

{

const char\* begin1 = str1;

while (\*begin2)

{

if (\*begin1 == \*begin2)

{

begin1++;

begin2++;

}

else //遇到不相等的情况就退出比较

break;

}

if(\*begin2 =='\0')//是因为str2为\0的原因跳出来的就返回地址

return str1;

}

begin2 = (char\*)str2;//重置begin2\_cpy

str1++;

}

return NULL;

}