实验报告：strcpy

危险的字符串复制

Author: Brandon Lin, Novembre 6th, 2023

内容

指针——向前奔跑的他并不知道身在何处

使用指针来遍历一个数组或者批量传入内容是一个非常愉快的经历，然而必须得承认这个操作带来一定的危险。我们通常都会初始化一个数组的最大内存，或者以 '\0' 标记字符串的末尾，但是当无脑使用 `ptrString++` 时，我们很可能已经超过了最大内存的边界，为此，我们需要人为数数；或者设置一个“边界检查站”，当指针遇到这个检查站时就停止向前继续奔跑了。

代码解释

我们以一个最大长度为 10 的字符串为例。首先判断传入字符串以及目标字符串地址的合法性。

1. 当输入一个长度小于 10 的字符，例如 `shortstr`，一切都没有问题
2. 当输入一个长度大于 10 的字符，例如 `longstr+longstr`，没有长度判断的 `strcpy` 函数停止工作（当然我觉得这也跟编译器有关吧，作为保护程序和内存的一种手段）；而有长度判断的 `ourstrncpy` 函数进行一个截断处理。
3. 【扩展】当提前对于字符串的后面 10 位进行初始化（这样就是有 20 位被初始化，但有可能这 10 位已经有其他用途，已经算是非法侵占空间了），即使字符串长度超过已经定义的数组的长度，依然正常读入。当然这真的不太好，所以不推荐这样操作。

备注：我登录不上服务器平台，基于本地 Mac 环境跑的结果应该是这样的，一次对应三个部分的内容：

```
We set the longest string length is : 10
****************************************
First str: shortstr
With the function strcpy, we obtain loc1: shortstr
With the function strncpy, we obtain loc2: shortstr
****************************************
Second str: longstr+longstr
With the function strcpy, we obtain loc1:
With the function strncpy, we obtain loc2: longstr+lo
****************************************
With initialization of the following (undefined) spaces after loc1, using the fu
nction strcpy, we obtain loc1: longstr+longstr
```

源码

```c
#include <stdio.h>

#define MAX_STRING 10

// 声明自定义的字符串拷贝函数
```

```c
char * strcpy(char * strDestination, const char * strSource);
char * ourstrncpy(char * strDestination, const char * strSource, int
strDestinationMaximumLength);

int main() {
    char loc1[MAX_STRING], loc2[MAX_STRING];
    char *ptrString1 = "shortstr";
    char *ptrString2 = "longstr+longstr";

    printf("We set the longest string length is : %d\n", MAX_STRING);
    printf("****************************************\n");

    printf("First str: shortstr\n");

    // 使用标准的 strcpy 函数复制字符串
    strcpy(loc1, ptrString1);

    // 使用自定义的 ourstrncpy 函数复制字符串，限制长度为 MAX_STRING
    ourstrncpy(loc2, ptrString1, MAX_STRING);

    printf("With the function strcpy, we obtain loc1: %s\n", loc1);
    printf("With the function strncpy, we obtain loc2: %s\n", loc2);

    printf("****************************************\n");
    printf("Second str: longstr+longstr\n");

    // 使用标准的 strcpy 函数复制较长的字符串
    strcpy(loc1, ptrString2);

    // 使用自定义的 ourstrncpy 函数复制较长的字符串，限制长度为 MAX_STRING
    ourstrncpy(loc2, ptrString2, MAX_STRING);

    printf("With the function strcpy, we obtain loc1: %s\n", loc1);
    printf("With the function strncpy, we obtain loc2: %s\n", loc2);

    // 将 loc1 中，以及其随后的空间的字符初始化为零
    char * ptrStringTest = loc1;
    for(int count=0; count<= MAX_STRING * 2; count++) {
        *ptrStringTest++ = 0;
    }

    printf("****************************************\n");

    // 使用标准的 strcpy 函数复制字符串，但在超出 MAX_STRING 长度后的字符未定义
    strcpy(loc1, ptrString2);
    printf("With initialization of the following (undefined) spaces after loc1,
using the function strcpy, we obtain loc1: %s\n", loc1);

    return 0;
}

// 自定义的字符串拷贝函数，复制一个字符串到另一个字符串
char * strcpy(char * strDestination, const char * strSource) {
    if (strDestination == NULL || strSource == NULL) {
        return NULL;
    }
    char *ptrDestination = strDestination;
```

```
    // 逐个字符复制源字符串到目标字符串，直到遇到字符串结束符 '\0'
    while ((*ptrDestination++ = *strSource++) != '\0') {
        ;
    }
    return strDestination;
}

// 自定义的字符串拷贝函数，复制一个字符串到另一个字符串，并限制最大长度
char * ourstrncpy(char * strDestination, const char * strSource, int
strDestinationMaximumLength) {
    if (strDestination == NULL || strSource == NULL) {
        return NULL;
    }
    char *ptrDestination = strDestination;

    // 逐个字符复制源字符串到目标字符串，直到遇到字符串结束符 '\0' 或达到最大长度
    for( int count = 0;
         (*ptrDestination++ = *strSource++) != '\0';
         count++) {
            if (count >= strDestinationMaximumLength) {
                *--ptrDestination = '\0';
                break;
            }
         }

    return strDestination;
}
```