­­­

# Memset实验报告

­­

姓名：白思雨

学号：2186123935

日期：2020年12月

# 实验：更有效的 Memset 代码实现

## 概述

### 1.1实验任务

### 

### 1.2结果综述

本实验参考PPT内容和提示实现了一个更为有效的memset函数，并且对不同n值都有较好的效果，相较原来提供的实现来说，运行效率大大提高。

## 实验过程及结果

### 2.1 目的

实现一个更有效的 memset 函数。

### 2.2 实验过程简述

**1. 查阅资料**

对现有的memset进行分析，它只是进行了简单的赋值，使用的也是int型数据，而实验要求是对不同n值都能够很好的执行并且高效执行，所以我首先阅读了PPT，复习了优化的相关知识，其次参考了网上其他人所写的memset函数，对之进行综合学习。

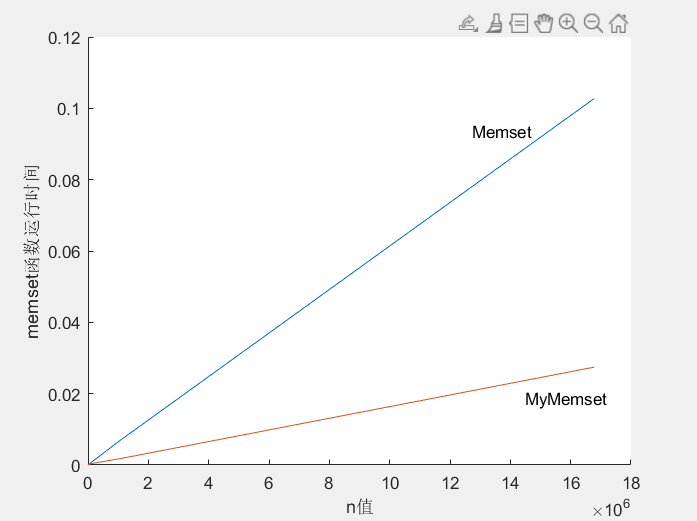
**2. 分析实行**

首先我们像原有函数一样使用计数器计数，并且在K整数倍时跳出(为了实现字单位读写)，由于我们实现第一个字的时候已经将它赋值完成，所以接着我们定义两个变量，一个是为了计算总的n减去K对K的倍数的数量，另一个是当不整除时n减去K对K的余数，后面我们把第一个按loop来进行字赋值，余数则继续进行字节赋值，因为它的赋值单位改变，因此可以提升该函数的运行效率。

本实验采用内置 clock() 函数来计算时间。

### 实验结果描述

随着n的增大，Memset函数的运行时间增长较快，My\_Memset运行时间增长较慢，且Memset函数的增长速度始终高于My\_Memset函数的增长速度。并且，对于不同n值My\_Memset函数均能成功执行。因此，我们的实验对memset函数的优化较为成功，以下是对于实验结果的图示。



## 总结体会与建议

通过这个实验的完成，我对优化的过程、原理，还有对内存分配都有了更深刻的认识，但在代码实现过程中也遇到了不少困难，希望以后的实践过程可以更加顺利，要多加练习。

## 实验代码

1. #include <stdio.h>
2. #include <stddef.h>
3. #include <stdint.h>
4. #include <stdlib.h>
5. #include <string.h>
6. #include <assert.h>
7. #include <time.h>
9. **void**\* Memset(**void** \*s, **int** c, **size\_t** n)
10. {
11. **size\_t** cnt = 0;
12. unsigned **char** \*schar = s;
13. **while** (cnt < n)
14. {
15. \*schar++ = (unsigned **char**) c;
16. cnt++;
17. }
18. **return** s;
19. }
21. **void**\* My\_Memset(**void** \*s, unsigned **long** cs, **size\_t** n)
22. {
23. **size\_t** K = **sizeof**(unsigned **long**);
24. **size\_t** cnt = 0;
25. unsigned **char** \*schar = s;
26. **while** (cnt < n)
27. {
28. **if** ((**size\_t**)schar % K == 0)
29. **break**;
30. \*schar++ = (unsigned **char**)cs;
31. cnt++;
32. }
34. unsigned **long** \*slong = (unsigned **long** \*)schar;
35. **size\_t** rest = n - cnt;
36. **size\_t** loop = rest / K;
37. **size\_t** tail = rest % K;
38. **for** (**size\_t** i = 0; i < loop; i++)
39. \*slong++ = cs;
41. schar = (unsigned **char** \*)slong;
42. **for** (**size\_t** i = 0; i < tail; i++)
43. \*schar++ = (unsigned **char**)cs;
44. **return** s;
45. }
47. #define N (1024\*1024LL)
48. #define INLINE inline \_\_attribute\_\_((always\_inline))
50. **char** src[2\*N] \_\_attribute\_\_((aligned(32)));
52. **int** main()
53. {
54. **clock\_t** start,finish;
55. start=clock();
56. Memset(src,0,**sizeof**(src));
57. //My\_Memset(src,'0',sizeof(src));
58. finish = clock();
59. printf("time=%lf\n", (**double**)(finish - start)/CLOCKS\_PER\_SEC);
60. **return** 0;
61. }