《计算机系统基础》Homework

HW3　存储器层次结构和存储保护

实验（一）　基于 cache 的存储访问

实验目的：通过实际程序的执行结果，了解程序访问局部性对带有 cache 的计算机系统性能的影响。
实验要求：在以下程序中修改或添加必要的语句（如添加计时函数等），以计算和打印主体程序段（即 for 循环段）的执行时间。要求分别给出程序代码（压缩包或截图）和结果（表格或代码运行结果截图）。
分别以“M=10，N=100 00000”；“M=10000，N=10000”；“M=100 00000，N=10”执行程序 A 和程序 B，以比较两种 for 循环段执行时间的长短。

程序段 A

```
short a[M][N];
assign-array-rows ( )
{
    int i, j;
    ……
     for    (i= 0; i<M; i++)
             for (j= 0; j<N; j++)
                    a[i][j]=0;
    ……
}
```

程序段 B

```
short a[M][N];
assign-array-cols ( )
{
    int i, j;
    ……
    for    (j= 0; j<N; j++)
             for (i=0; i<M; i++)
                    a[i][j]=0;
     ……
 }
```

实验报告：

1. 分别给出在 Windows 和 Linux 系统中的执行结果。
2. 修改程序使 a 作为非静态局部变量并分配在栈区，分别给出在 Windows 和 Linux 系统中的执行结果。(注：windows 和 linux 对栈分配大小有一定限制，需要手动修改)
3. 修改程序使 a 作为非静态局部变量并分配在堆区，分别给出在 Windows 和 Linux 系统中的执行结果。
4. 对上述执行结果进行分析，说明局部数据块大小、数组访问顺序等和执行时间之间的关系。

实验（二） 存储保护

实验目的：通过实际程序的执行结果，了解程序执行时访问违例的检测和处理过程。

实验要求：执行以下程序，通过 gdb 调试找到发生异常的指令以及发生访问违例的存储单元地址。

主要程序段如下：

```
int sum(int a[ ], unsigned len)
{
    int   i，sum = 0;
    for (i = 0; i <= len–1; i++)
          sum += a[i];
    return sum;
}
```

实验报告：

1. 使用 len=0 调用 sum 函数，分别给出在 Windows 和 Linux 系统中的执行结果。
2. 在 Linux 系统中，用 gdb 调试工具，通过设置正确的断点、显示通用寄存器的内容等手段，确定发生异常的指令，并指出发生的是什么异常，以及发生访问违例的存储单元地址。
3. 从 CPU 检测到异常到屏幕中出现“Segment fault”的整个过程中，CPU 和操作系统各做了哪些事情？