C++的文件

在竞赛中，遇到大数据时，往往读文件成了程序运行速度的瓶颈，需要更快的读取方式。相信几乎所有的 C++学习者都在 cin 机器缓慢的速度上栽过跟头，于是从此以后发誓不用 cin 读数据。还有人说 Pascal 的 read 语句的速度是 C/C++中 scanf 比不上的，C++选手只能干着急。难道 C++真的低 Pascal 一等吗？答案是不言而喻的。一个进阶的方法是把数据一下子读进来，然后再转化字符串，这种方法传说中很不错，但具体如何从没试过，因此今天就索性把能想到的所有的读数据的方式都测试了一边，结果是惊人的。

竞赛中读数据的情况最多的莫过于读一大堆整数了，于是我写了一个程序，生成一千万个随机数到 data.txt 中，一共 55MB。然后我写了个程序主干计算运行时间，代码如下：

View Code CPP

```cpp
#include <ctime>
int main()
{
        int start = clock();
        //DO SOMETHING
        printf("%.3lf\n",double(clock()-start)/CLOCKS_PER_SEC);
}
```

最简单的方法就算写一个循环 scanf 了，代码如下：

View Code CPP

```cpp
const int MAXN = 10000000;

int numbers[MAXN];

void scanf_read()
{
        freopen("data.txt","r",stdin);
```

```
8         for (int i=0;i<MAXN;i++)
9                 scanf("%d",&numbers[i]);
10 }
```

可是效率如何呢？在我的电脑 Linux 平台上测试结果为 2.01 秒。接下来是 cin，代码如下

View Code CPP

```
1  const int MAXN = 10000000;
2
3  int numbers[MAXN];
4
5  void cin_read()
6  {
7         freopen("data.txt","r",stdin);
8         for (int i=0;i<MAXN;i++)
9                 std::cin >> numbers[i];
10 }
```

出乎我的意料，cin 仅仅用了 6.38 秒，比我想象的要快。cin 慢是有原因的，其实默认的时候，cin 与 stdin 总是保持同步的，也就是说这两种方法可以混用，而不必担心文件指针混乱，同时 cout 和 stdout 也一样，两者混用不会输出顺序错乱。正因为这个兼容性的特性，导致 cin 有许多额外的开销，如何禁用这个特性呢？只需一个语句 std::ios::sync_with_stdio(false);，这样就可以取消 cin 于 stdin 的同步了。程序如下：

View Code CPP

```
1  const int MAXN = 10000000;
2
3  int numbers[MAXN];
4
5  void cin_read_nosync()
6  {
```

```
7           freopen("data.txt","r",stdin);
8           std::ios::sync_with_stdio(false);
9           for (int i=0;i<MAXN;i++)
10                  std::cin >> numbers[i];
11  }
```

取消同步后效率究竟如何？经测试运行时间锐减到了 2.05 秒，**与 scanf 效率相差无几了！**

有了这个以后可以放心使用 cin 和 cout 了。

接下来让我们测试一下读入整个文件再处理的方法，首先要写一个字符串转化为数组的函数，代码如下

View Code CPP

```
1  const int MAXS = 60*1024*1024;
2  char buf[MAXS];
3
4  void analyse(char *buf,int len = MAXS)
5  {
6           int i;
7           numbers[i=0]=0;
8           for (char *p=buf;*p && p-buf<len;p++)
9                   if (*p == ' ')
10                          numbers[++i]=0;
11                  else
12                          numbers[i] = numbers[i] * 10 + *p - '0';
13  }
```

把整个文件读入一个字符串最常用的方法是用 fread，代码如下：

View Code CPP

```
1  const int MAXN = 10000000;
2  const int MAXS = 60*1024*1024;
3
```

```cpp
int numbers[MAXN];
char buf[MAXS];

void fread_analyse()
{
        freopen("data.txt","rb",stdin);
        int len = fread(buf,1,MAXS,stdin);
        buf[len] = '\0';
        analyse(buf,len);
}
```

上述代码有着惊人的效率，经测试读取这 10000000 个数只用了 0.29 秒，效率提高了几乎 10 倍！掌握着种方法简直无敌了，不过，我记得 fread 是封装过的 read，如果直接使用 read，是不是更快呢？代码如下：

View Code CPP

```cpp
const int MAXN = 10000000;
const int MAXS = 60*1024*1024;

int numbers[MAXN];
char buf[MAXS];

void read_analyse()
{
        int fd = open("data.txt",O_RDONLY);
        int len = read(fd,buf,MAXS);
        buf[len] = '\0';
        analyse(buf,len);
}
```

测试发现运行时间仍然是 0.29 秒，可见 read 不具备特殊的优势。到此已经结束了吗？不，我可以调用 Linux 的底层函数 mmap，这个函数的功能是将文件映射到内存，是所有读文件方法都要封装的基础方法，直接使用 mmap 会怎样呢？代码如下：

View Code CPP

```cpp
const int MAXN = 10000000;
const int MAXS = 60*1024*1024;

int numbers[MAXN];
char buf[MAXS];
void mmap_analyse()
{
        int fd = open("data.txt",O_RDONLY);
        int len = lseek(fd,0,SEEK_END);
        char *mbuf = (char *)
mmap(NULL,len,PROT_READ,MAP_PRIVATE,fd,0);
        analyse(mbuf,len);
}
```

经测试，运行时间缩短到了 0.25 秒，效率继续提高了 14%。到此为止我已经没有更好的方法继续提高读文件的速度了。回头测一下 Pascal 的速度如何？结果**令人大跌眼镜**，居然运行了 2.16 秒之多。程序如下：

View Code PASCAL

```pascal
const
        MAXN = 10000000;
var
        numbers :array[0..MAXN] of longint;
        i :longint;
begin
        assign(input,'data.txt');
```

```
8           reset(input);
9           for i:=0 to MAXN do
10                  read(numbers[i]);
11 end.
```

为确保准确性，我又换到 Windows 平台上测试了一下。结果如下表：

方法/平台/时间(秒)	Linux gcc	Windows mingw	Windows VC2008
scanf	2.010	3.704	3.425
cin	6.380	64.003	19.208
cin 取消同步	2.050	6.004	19.616
fread	0.290	0.241	0.304
read	0.290	0.398	不支持
mmap	0.250	不支持	不支持
Pascal read	2.160	4.668	

从上面可以看出几个问题

1. Linux 平台上运行程序普遍比 Windows 上快。
2. Windows 下 VC 编译的程序一般运行比 MINGW（MINimal Gcc for Windows）快。
3. VC 对 cin 取消同步与否**不敏感**，前后效率相同。反过来 MINGW 则**非常敏感**，前后效率相差 8 倍。
4. read 本是 linux 系统函数，MINGW 可能采用了某种模拟方式，read 比 fread 更慢。
5. Pascal 程序运行速度实在令人不敢恭维。

希望此文能对大家有所启发，欢迎与我继续讨论。