

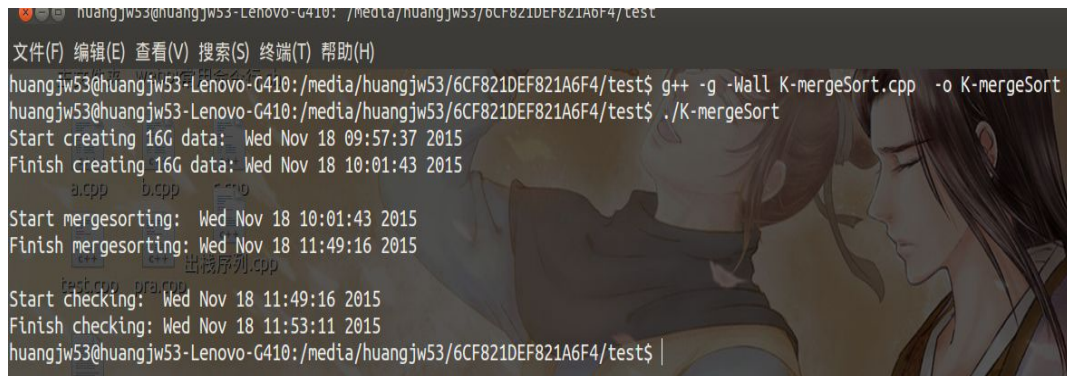
文件归并排序

对 2G 的数据量排序，因为每个 long int 为 8 个字节，所以该二进制文件共 16G(RandomData)，先调用 rand() 随机生成 20 亿个 long int，因为程序的运行环境为 Linux，rand 生成的为 int，而在 window 下生成的为 short，故随机生成 $20 \text{ 亿} \times 8 / \text{sizeof}(\text{rand}())$ 个数，读取和写入的时候一次读取 8 个字节即可。

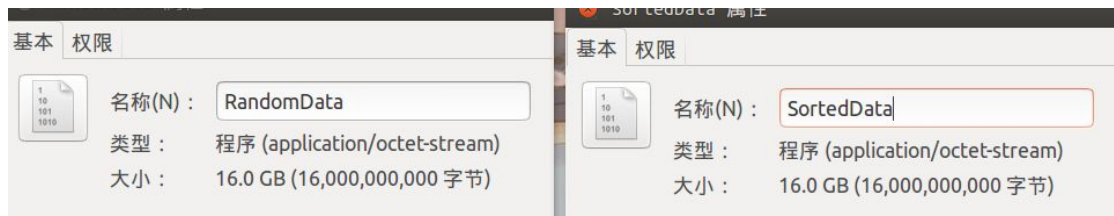
一开始完全用文件操作，不开数组(merge_sort.cpp)，测试了 10 万，100 万，1000 万的数据量，发现时间几乎每次成 10 倍增长，估计排 20 亿个数纯文件操作需 40 个小时，甚至更久。。。。。

于是改进和优化算法，文件的读写速度远小于内存的读写速度，故开了个 1000 万的 long 数组进行内部排序(K-mergeSort.cpp)，每次从文件中读取 1000 万个数存进数组，然后对该数组快排，再写进另一个文件，通过以上方法将 16G 原文件分成 200 个有序的子文件，然后再两个两个子文件合并成一个更大的有序的文件，不断循环合并，直至只剩下一个文件。

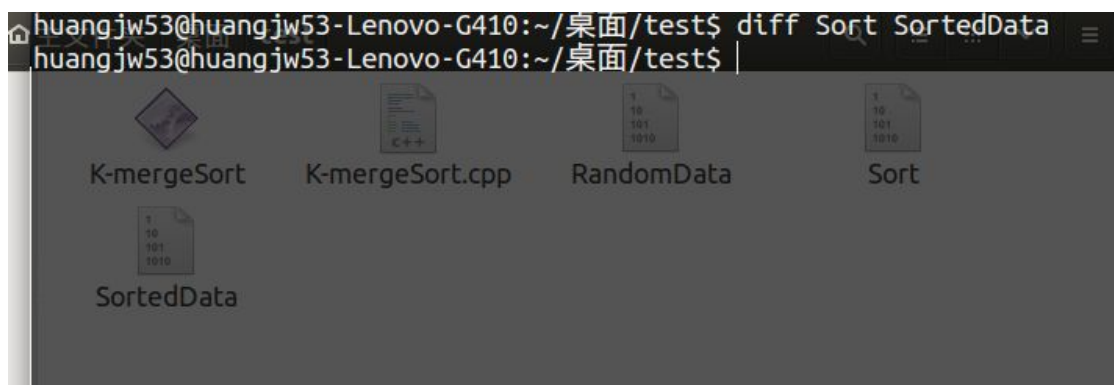
经运行，优化后 2 个小时即可排完 16G 的文件



```
huangjw53@huangjw53-Lenovo-G410: /media/huangjw53/6CF821DEF821A6F4/test$ g++ -g -Wall K-mergeSort.cpp -o K-mergeSort
huangjw53@huangjw53-Lenovo-G410: /media/huangjw53/6CF821DEF821A6F4/test$ ./K-mergeSort
Start creating 16G data: Wed Nov 18 09:57:37 2015
Finish creating 16G data: Wed Nov 18 10:01:43 2015
Start mergesorting: Wed Nov 18 10:01:43 2015
Finish mergesorting: Wed Nov 18 11:49:16 2015
Start checking: Wed Nov 18 11:49:16 2015
Finish checking: Wed Nov 18 11:53:11 2015
huangjw53@huangjw53-Lenovo-G410: /media/huangjw53/6CF821DEF821A6F4/test$
```



检验 16G 文件是否有序只能采用遍历，判断后一个数是否大于前一个数，故另外生成 1000 万个数 RandomData，存进数组快排后写入文件 Sort，再对 RandomData 使用 K-mergeSort，同样分成 200 份，每份 5 万个数，归并生成 SortedData，使用 Linux 的文件比较，经比较，两者完全相同。



今晚下课回到宿舍，在 windows 下运行了该程序，发现有些不严谨的地方编译错误,改完编译错误过后，发现排序结果是错的，在室友电脑运行甚至崩溃了，调试后发现，windows 下 rand()产生 short 没错，可是 sizeof(rand())却是 4，而且 long 也只占 4 个字节，这都是之前完全没想到的，才会使在 Linux 下正确的程序得到错误的结果甚至崩溃，为了适应不同的系统，将代码改为排 20 亿个 double，这样每个数都是 8 字节，已经在 windows 和 Linux 分别测试了新的代码，结果均正确。