

---

# 操作系统实验二

快速排序问题

---

董凯杰

无 34

2013010572

2015 年 12 月 4 日

# 目录

一、 实验目的	2
二、 实验题目	2
三、 问题描述	2
四、 实现要求	2
五、 实验设计	3
六、 实验结果	7
七、 思考题	8
八、 实验总结	8

## 一、 实验目的

- (1) 通过对进程间高级通信问题的编程实现，加深理解进程间高级通信的原理；
- (2) 对 Windows 或 Linux 涉及的几种高级进程间通信机制有更进一步的了解；
- (3) 熟悉 Windows 或 Linux 中定义的与高级进程间通信有关的函数。

## 二、 实验题目

快速排序问题

## 三、 问题描述

对于有 1,000,000 个乱序数据的数据文件执行快速排序。

## 四、 实现要求

1. 首先产生包含 1,000,000 个随机数（数据类型可选整型或者浮点型）的数据文件；
2. 每次数据分割后产生两个新的进程（或线程）处理分割后的数据，每个进程（线程）处理的数据小于 1000 以后不再分割（控制产生的进程在 20 个左右）；
3. 线程（或进程）之间的通信可以选择下述机制之一进行：
  - 管道（无名管道或命名管道）
  - 消息队列
  - 共享内存
4. 通过适当的函数调用创建上述 IPC 对象，通过调用适当的函数调用实现数据的读出与写入；
5. 需要考虑线程（或进程）间的同步；
6. 线程（或进程）运行结束，通过适当的系统调用结束线程（或进程）。

## 五、 实验设计

### 设计思路

采用文件映射机制，用户进程可以将整个文件映射为进程虚拟地址空间的一部分来加以访问。通过不停的由母进程生成子进程来对同一块数组内存进行排序修改，以实现多进程之间共享内存。

### 程序框架

```
//快速排序函数
void qsort(int *arr, int left, int right, int &len1, int &len2, int &position)
{
    if (left >= right) return;
    int leftTemp = left;
    int rightTemp = right;
    int key = arr[left];
    while (left < right)
    {
        while (left < right && arr[right] >= key) right--;
        arr[left] = arr[right];
        while (left < right && arr[left] <= key) left++;
        arr[right] = arr[left];
    }
    arr[left] = key;
    len1 = left - leftTemp;
    len2 = rightTemp - left;
    position = left;

    //如果当前处理的数据长度小于1000，则继续递归，
    //直到完成当前长度的数据的排序
    //否则直接返回，由母线程进一步生成两个子线程进行排序
    if (rightTemp - leftTemp + 1 <= 1000)
    {
        qsort(arr, leftTemp, left - 1, len1, len2, position);
        qsort(arr, left + 1, rightTemp, len1, len2, position);
    }
}
```

```
DWORD WINAPI Qsort(LPVOID lpPara) //快速排序线程
{
    //线程参数，每个线程需要处理的数组长度，开始位置，结束位置
    ArrInfo *arrinfo = (ArrInfo *)lpPara;
    //打开已创建的文件映射对象
    HANDLE hFileMap = OpenFileMapping(FILE_MAP_ALL_ACCESS,
                                      FALSE, L"data");
    //将文件视图映射到本进程的地址空间
    VOID *hMap = MapViewOfFile(hFileMap,
                               FILE_MAP_ALL_ACCESS, 0, 0, 0);
    //获得各个线程共享的数组首地址
    int *arr1 = (int *)hMap;

    WaitForSingleObject(Qsort_mutex, INFINITE);
    int left = arrinfo->start;
    int right = arrinfo->end;
    int len1, len2, position;
    //对当前长度的数组进行快速排序
    qsort(arr1, left, right, len1, len2, position);
    ReleaseMutex(Qsort_mutex);

    ArrInfo arrinfo1, arrinfo2;
    arrinfo1.start = arrinfo->start;
    arrinfo1.end = position - 1;
    arrinfo1.len = len1;
    arrinfo2.start = position + 1;
    arrinfo2.end = arrinfo->end;
    arrinfo2.len = len2;
    //创建第一个线程
    if (arrinfo1.len > 1000)
    {
        HANDLE hQsort1 = CreateThread(NULL, 0,
                                       (LPTHREAD_START_ROUTINE)Qsort, &arrinfo1, 0, NULL);
        WaitForSingleObject(hQsort1, INFINITE);
        CloseHandle(hQsort1);
    }
}
```

```
else
    //如果数据小于1000, 则不在继续分割产生进程, 而是完成该段数据快速排序
    qsort(arr1, arrinfo1.start, arrinfo1.end, len1, len2, position);
//创建第二个线程
if (arrinfo2.len > 1000)
{
    HANDLE hQsort2 = CreateThread(NULL, 0,
    (LPTHREAD_START_ROUTINE)Qsort, &arrinfo2, 0, NULL);
    WaitForSingleObject(hQsort2, INFINITE);
    CloseHandle(hQsort2);
}
else
    //如果数据小于1000, 则不在继续分割产生进程, 而是完成该段数据快速排序
    qsort(arr1, arrinfo2.start, arrinfo2.end, len1, len2, position);

UnmapViewOfFile(hMap);
CloseHandle(hFileMap);
return 0;
}
int main()//主线程
{
    //从文件中读取数据
    ifstream file ("input.txt");
    int num = 1000000;
    vector<int> a;
    int x;
    for (int i=0; i<num; i++)
    {
        file >> x;
        a.push_back(x);
    }
    //创建排序互斥量
    Qsort_mutex = CreateMutex(NULL, FALSE, NULL);
    //创建文件映射对象
```

```

HANDLE hFileMap = CreateFileMapping(INVALID_HANDLE_VALUE,
    NULL,PAGE_READWRITE,0,0x1000000,L"data");
//将文件视图映射到本进程的地址空间
VOID *hMap = MapViewOfFile(hFileMap,FILE_MAP_ALL_ACCESS,0,0,0);
//将数组首地址关联映射地址空间的首地址，使它们指向同一块内存
memcpy(hMap, &a[0], num * sizeof(int));

//快排线程参数
ArrInfo arrinfo;
arrinfo.start = 0;
arrinfo.end = num - 1;
arrinfo.len = num;
//创建快排主线程
HANDLE hQsort = CreateThread(NULL, 0, (
    LPTHREAD_START_ROUTINE)Qsort, &arrinfo, 0, NULL);
WaitForSingleObject(hQsort, INFINITE);
CloseHandle(hQsort);
//将结果写回文件
int *b = (int *)hMap;
ofstream output("output.txt");
for (int i = 0; i < 1000000; i++)
    output << b[i] << '\n';
//打印前100个数据，以便查验
//for (int i=0; i<100; i++)
//{
//    cout<< b[i]<<endl;
//}
UnmapViewOfFile(hMap);
CloseHandle(hFileMap);
}

```

## 说明

Qsort\_mutex（排序互斥量）保证在同一时间只有一个线程能对共享内存进行操作。

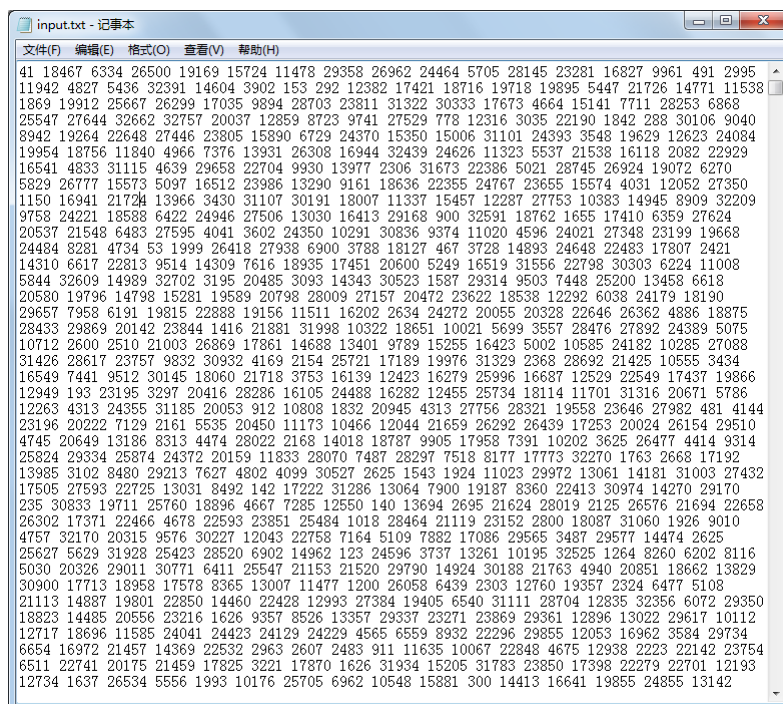
ArrInfo \*arrinfo（线程参数）保证每个线程处理不同位置的数据。

线程的创建方式为单向创建，即母线程先创建第一个线程，然后直到第一个线程执行完毕后，才会创建第二个进程。因此，同时工作的线程个数与递归层数相近，大致为

20 个。

## 六、 实验结果

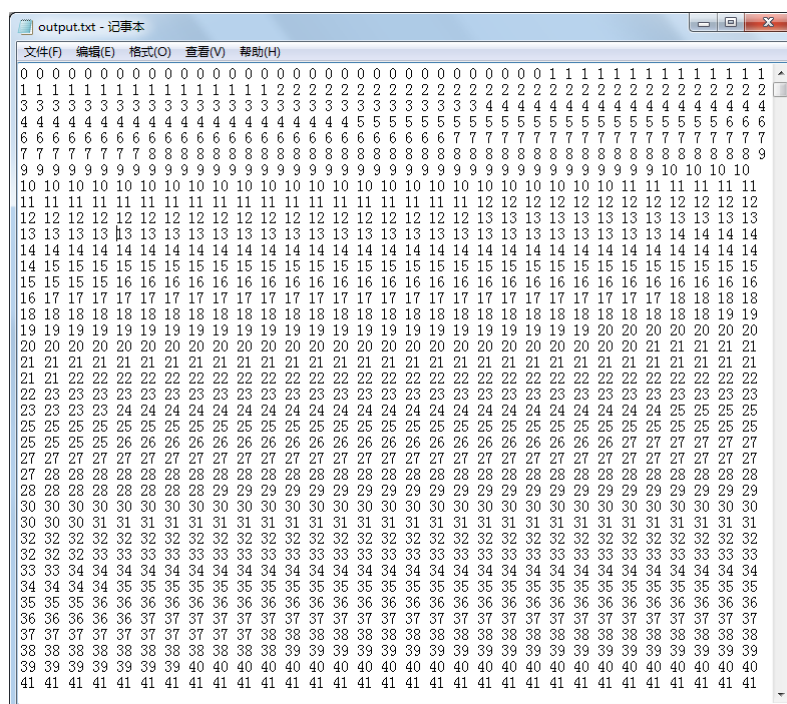
排序前部分数据截图：



```
input.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
41 18467 6334 26500 19169 15724 11478 29358 26962 24464 5705 28145 23281 16827 9961 491 2995
11942 4827 5436 32391 14604 3902 153 292 12382 17421 18716 19718 19895 5447 21726 14771 11538
1869 19912 25667 26299 17035 9894 28703 23811 31322 30333 17673 4664 15141 7711 28253 6868
25547 27644 32662 32757 20037 12859 8723 9741 27529 778 12316 3035 22190 1842 288 30106 9040
8942 19264 22648 27446 23805 15890 6729 24370 15350 15006 31101 24393 3548 19629 12623 24084
19954 18756 11840 4966 7376 13931 26308 16944 32439 24626 11323 5537 21538 16118 2082 22929
16541 4833 31115 4639 29658 22704 9930 13977 2306 31673 22386 5021 28745 26924 19072 6270
5829 26777 15573 5097 16512 23986 13290 9161 18636 22355 24767 23655 15574 4031 12052 27350
1150 16941 21724 13966 3430 31107 30191 18007 11337 15457 12287 27753 10383 14945 8909 32209
9758 24221 18588 6422 24946 27506 13030 16413 29168 900 32591 18762 1655 17410 6359 27624
20537 21548 6483 27595 4041 3602 24350 10291 30836 9374 11020 4596 24021 27348 23199 19668
24484 8281 4734 53 1999 26418 27938 6900 3788 18127 467 3728 14893 24648 22483 17807 2421
14310 6617 22813 9514 14309 7616 18935 17451 20600 5249 16519 31556 22798 30303 6224 11008
5844 32609 14989 32702 3195 20485 3093 14343 30523 1587 29314 9503 7448 25200 13458 6618
20580 19796 14798 15281 19589 20798 28009 27157 20472 23622 18538 12292 6038 24179 18190
29657 7958 6191 19815 22888 19156 11511 16202 2634 24272 20055 20328 22646 26362 4886 18875
28433 29869 20142 23844 1416 21881 31998 10322 18651 10021 5699 3557 28476 27892 24389 5075
10712 2600 2510 21003 26869 17861 14688 13401 9789 15255 16423 5002 10585 24182 10285 27088
31426 28617 23757 9832 30932 4169 2154 25721 17189 19976 31329 2368 28692 21425 10555 3434
16549 7441 9512 30145 18060 21718 3753 16139 12423 16279 25996 16687 12529 22549 17437 19866
12949 193 23195 3297 20416 28286 16105 24488 16282 12455 25734 18114 11701 31316 20671 5786
12263 4313 24355 31185 20053 912 10808 1832 20945 4313 27756 28321 19558 23646 27982 481 4144
23196 20222 7129 2161 5535 20450 11173 10466 12044 21659 26292 26439 17253 20024 26154 29510
4745 20649 13186 8313 4474 28022 2168 14018 18787 9905 17958 7391 10202 3625 26477 4414 9314
25824 29334 25874 24372 20159 11833 28070 7487 28297 7518 8177 17773 32270 1763 2668 17192
13985 3102 8480 29213 7627 4802 4099 30527 2625 1543 1924 11023 29972 13061 14181 31003 27432
17505 27593 22725 13031 8492 142 17222 31286 13064 7900 19187 8360 22413 30974 14270 29170
235 30833 19711 25760 18896 4667 7285 12550 140 13694 2695 21624 28019 2125 26576 21694 22658
26302 17371 22466 4678 22593 23851 25484 1018 28464 21119 23152 2800 18087 31060 1926 9010
4757 32170 20315 9576 30227 12043 22758 7164 5109 7882 17086 29565 3487 29577 14474 2625
25627 5629 31928 25423 28520 6902 14962 123 24596 3737 13261 10195 32525 1264 8260 6202 8116
5030 20326 29011 30771 6411 25547 21153 21520 29790 14924 30188 21763 4940 20851 18662 13829
30900 17713 18958 17578 8365 13007 11477 1200 26058 6439 2303 12760 19357 2324 6477 5108
21113 14887 19801 22850 14460 22428 12993 27384 19405 6540 31111 28704 12835 32356 6072 29350
18823 14485 20556 23216 1626 9357 8526 13357 29337 23271 23869 29361 12896 13022 29617 10112
12717 18696 11585 24041 24423 24129 24229 4565 6559 8932 22296 29855 12053 16962 3584 29734
6654 16972 21457 14369 22532 2963 2607 2483 911 11635 10067 22848 4675 12938 2223 22142 23754
6511 22741 20175 21459 17825 3221 17870 1626 31934 15205 31783 23850 17398 22279 22701 12193
12734 1637 26534 5556 1993 10176 25705 6962 10548 15881 300 14413 16641 19855 24855 13142
```

排序后部分数据截图：





## 七、 思考题

1. 你采用了你选择的机制而不是另外的两种机制解决该问题，请解释你做出这种选择的理由。

考虑到是对数组的排序，直接对内存进行操作比较方便，因此选择共享内存的方式。每个线程独立操作，修改部分数据后即可形成最后的排序结果。

2. 你认为另外的两种机制是否同样可以解决该问题？如果可以请给出你的思路；如果不能，请解释理由。

都可以解决该问题。使用管道的方式，需要母线程对数组序列进行类似的二分操作，将这两部分通过管道传输给两个子线程。使用消息队列，其实现方式基本与管道所类似。

## 八、 实验总结

本次实验难度并不是很大，尤其是有了第一次实验的基础，对于线程、信号量等库函数更加熟悉，同时对于线程的执行方式、顺序也有了比较深刻的体会。

当然一开始也在文件映射的存储方式使用上遇到一些困难，总是会读入乱码，因此针对这个问题，在请教马老师之后也相应的做了一些修改，跳过了这一比较繁琐的环

节。

总之，通过前两次有关进程的实验，让我对进程有了彻底的理解，对于书本上以及课上所讲内容有了更感性的认识，收获还是非常之大。