**Qsort快速排序**

Qsort简介：

快速排序是单线程最快的排序，但是多线程的情况下是多线程的希尔排序最快

#include <stdlib.h>

void qsort(

void \*buf, 第一个参数，内存首地址

size\_t num, 第二个参数排序的个数

size\_t size, 第三个参数要排序的元素类型大小

int (\*compare)(const void \*, const void \*)第四个参数就是比大小的函数

);

功能：

对buf 指向的数据(包含num 项,每项的大小为size)进行快速排序。

如果函数compare 的第一个参数小于第二个参数，返回负值；如果等于返回零值；如果大于返回正值。

函数对buf 指向的数据按升序排序

1.Qsort排序基本数据类型int：

int a[1000];

qsort(a, sizeof(a) / sizeof(a[0]), sizeof(int), compare);

int compare(void\* p1, void\* p2)//参数必须是void\*

{

int\* px1 = p1;

int\* px2 = p2;

if (\*px1 == \*px2)

{

return 0;//代表相等

}

else if (\*px1 < \*px2)

{

return -1;//-1代表小于

}

else

{

return 1;//代表大于

}

}

排序的应用：

网盘急速上传:先再服务器上查找文件是否存在，如果服务器上已经有这份文件了，就不用上传了

2.Qsort排序字符串：

char \* str[10] = { "abc", "abd", "bcf", "123", "456", "qwe", "ABC", "ABF", "xyz", "X" };//指针数组

qsort(str, sizeof(str) / sizeof(str[0]), sizeof(str[0]), compare);

//strcmp，不仅可以判定相等，还可以用于排序

int compare(void\* p1, void\* p2)

{

int\* px1 = p1;

int\* px2 = p2;

return strcmp(px1, px2);

}

3.Qsort排序自定义结构体类型

struct info

{

int id;// 4

char str[50];//50

//字节对齐56 要能整除最宽基本数据类型double 8

};

int compare\_by\_id(void\* p1, void\* p2)//依照id排序

{

struct info\* px1 = p1;

struct info\* px2 = p2;

//简单的返回就可以确定

return (px1->id - px2->id);

/\*

if (px1->id == px2->id)

{

return 0;

}

else if (px1->id < px2->id)

{

return -1;//-1代表小于

}

else if (px1->id > px2->id)

{

return 1;//代表大于

}

\*/

}

int compare\_by\_name(void\* p1, void\* p2)//依照name排序

{

struct info\* px1 = p1;

struct info\* px2 = p2;

return strcmp(px1->str, px2->str);

}

int main(void)

{

struct info in[] = {

{11, "chengzhi01"},

{22, "sadfsdfwe"},

{3, "vertsfasf"},

{4, "asffsdfsdf"}

};

printf("%d %d\n\n", sizeof(struct info), sizeof(in[0]));//两种表示都对

//sort by id

qsort(in, sizeof(in) / sizeof(struct info), sizeof(struct info), compare\_by\_id);

for (int i = 0; i < 4; i++)

{

printf("%d %s\n", in[i].id, in[i].str);

}

printf("\n\n\n");

//sort by name

qsort(in, sizeof(in) / sizeof(in[0]), sizeof(in[0]), compare\_by\_name);

for (int i = 0; i < 4; i++)

{

printf("%d %s\n", in[i].id, in[i].str);

}

system("pause");

return 0;

}

4.简化比较函数

都可以通过一行代码解决

int compare\_by\_id(void\* p1, void\* p2)

{

return (((struct info\*)p1)->id - ((struct info\*)p2)->id);

}

int compare\_by\_name(void\* p1, void\* p2)

{

return strcmp(((struct info\*)p1)->str, ((struct info\*)p2)->str);

}