Project 4　Multithreaded sorting application

程式說明

這次的sort我使用bubble sort,因為簡單方便，而且需要排序的數量很少，所以沒有什麼效率上的問題。

Struct的設計

```
typedef struct
{
        int len;
        int number[8];
}numArray;

typedef struct
{
        numArray *numArr[2];
        int numSort[16];
}numCollect;
```

bubble sort要用到的交換

```
void swap(int *n1,int *n2)
{
        int tmp;
        tmp = *n1;
        *n1 = *n2;
        *n2 = tmp;
}
```

將原Array分成兩個比較小的array做處理

```
void* multiSort(void* data)
{
        printf("Before Sorting\n");
        numArray *num = (numArray*)data;
        int i,j;
        for(i=0;i<num->len;i++) printf("%d ",num->number[i]);
        puts("");
        for(i=0;i<num->len;i++)
                for(j=i+1;j<num->len;j++)
                        if(num->number[i] > num->number[j])
                                swap(&num->number[i],&num->number[j]);
        printf("After Sorting\n");
        for(i=0;i<num->len;i++) printf("%d ",num->number[i]);
        puts("\n");
        pthread_exit(NULL);
}
```

這個thread只處理收到的int array,merge的功能則是交給別的thread處理。

merge用的thread

```
void* mergeNumArr(void* data)
{
        printf("merge now\n");
        numCollect *nc = (numCollect*)data;
        int i=0,j=0,now=0;
        int len1 = nc->numArr[0]->len;
        int len2 = nc->numArr[1]->len;

        while(1)
        {
                if(i < len1 && j < len2)
                        nc->numSort[now++] = (nc->numArr[0]->number[i] < nc->nu
mArr[1]->number[j]?nc->numArr[0]->number[i++]:nc->numArr[1]->number[j++]);
                else if(i < len1)
                        nc->numSort[now++] = nc->numArr[0]->number[i++];
                else if(j < len2)
                        nc->numSort[now++] = nc->numArr[1]->number[j++];
                else
                        break;
        }
        pthread_exit(NULL);
}
```

將兩個比較小的**array**重新合併成完整大小並且排序過的**array**。
由於兩個**array**都已經排序過，因此從頭開始比較即可。

結果展示

```
user@instant-contiki:~/Desktop/os/hw2$ ./multiSort.out
Start MultiSorting Algorithm
Input Amount of Numbers <= 16: 16
Input 16 number: 23 45 12 16 23 98 76 84 16 84 96 74 73 19 92 99
Before Sorting
16 84 96 74 73 19 92 99
After Sorting
16 19 73 74 84 92 96 99

Before Sorting
23 45 12 16 23 98 76 84
After Sorting
12 16 23 23 45 76 84 98

merge now
After Merge
12 16 16 19 23 23 45 73 74 76 84 84 92 96 98 99
End MultiSorting Algorithm
```