Operating System Project 3

潘梓丞 517030910349

The goal of the project is to sort using
The porject was done by VM VirtualBox 5.2.18
The code are written by C and the library needed will be shown in code

idea

Divide the $args$ waiting for sort into two parts, using two pthreads to sort each part independently. Then wait for them to complete, then merge the two parts and get the result.

code

```c
int main(void)
{
        int count=0,i;
        char input;
        while(1){
                input=getchar();
                if (input=='\n')
                        break;
                num[count]=input-'0';
                count++;
        }
    pthread_t tid;
    pthread_attr_t attr;
    pthread_attr_init(&attr);
    int left_right[2] = {0, count};
    pthread_create(&tid, &attr, quicksort, left_right);
    pthread_join(tid, NULL);

    for (i = 0; i < count - 1; ++i)
        printf("%d ", num[i]);
    printf("%d\n", num[count - 1]);

    return 0;
}

void *quicksort(void *left_right)
{
    int left = ((int *)left_right)[0], right = ((int *)left_right)[1];
    if (left + 1 >= right)
        return NULL;
    if (left + 2 == right) {
        if (num[left] > num[left + 1]) {
            int temp = num[left];
            num[left] = num[left + 1];
            num[left + 1] = temp;
        }
        return NULL;
    }

    int size = right - left, p = left + random() % size;
    int temp = num[p], pivot = temp;
    num[p] = num[right - 1];
    int j;
    for (p = j = left; j < right - 1; ++j)
        if (num[j] < pivot) {
            temp = num[j];
            num[j] = num[p];
            num[p++] = temp;
        }
    num[right - 1] = num[p];
    num[p] = pivot;
```

```c
int main()
{
    array = malloc(40* sizeof(int));
    res = malloc(40* sizeof(int));
    int s;
    for(s = 0; s < 40; s++)
    {
        array[s] = rand()%100;
        //printf("%d ", array[i]);
    }

    pthread_t sortingThreads[2];

    struct size_ sizes[2];
    sizes[0].low = 0;
    sizes[0].high = 19;
    sizes[1].low =20;
    sizes[1].high = 39;
    int k;
    for (k=0; k<2; k++)
    {
        pthread_create(&sortingThreads[k], NULL, sortThread, (void *)&sizes[k]);

    }

    /*
    pthread_create(&sortingThreads[0], NULL, sortThread1, NULL);
    pthread_create(&sortingThreads[1], NULL, sortThread2, NULL);
    */
    int j;
    for (j = 0; j < 2; j++)
        pthread_join(sortingThreads[j], NULL);



    pthread_t mergingThread;
    pthread_create(&mergingThread, NULL, merge, NULL);
    pthread_join(mergingThread, NULL);
    int i;
    for (i=0; i<40; i++) printf("%d ", res[i]);
          printf("\n");


    return 0;
}
```

```
void sort(int a, int b)
{
    int i,j;
    for(i = a; i < b; i++)
    {
        for(j = a+1; j < b + a - i + 1;j++)
        {
            if(array[j] < array[j-1])
            {
                int tmp = array[j];
                array[j] = array[j-1];
                array[j-1] = tmp;
            }
        }
    }
}
void* sortThread(void *arg)
{
    struct size_* s = (struct size_ *) arg;
    sort(s->low,s->high);

}



void* merge(void *arg)
{
    int idx1 = 0;
    int idx2 = 20;
    int cnt = 0;
    while(idx1 <= 19 || idx2 <= 39)
    {
        if(idx1 <= 19 && idx2 <= 39)
        {
            if(array[idx1] > array[idx2])
            {
                res[cnt++] = array[idx2++];
            }
            else
            {
                res[cnt++] = array[idx1++];
            }
        }
        else
        {
            if(idx1 <= 19)
            {
                while(cnt <= 39)
                {
                    res[cnt++] = array[idx1++];
                }
```