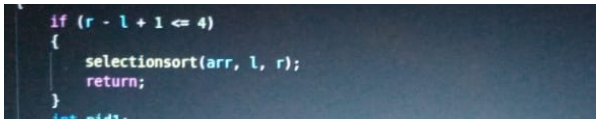
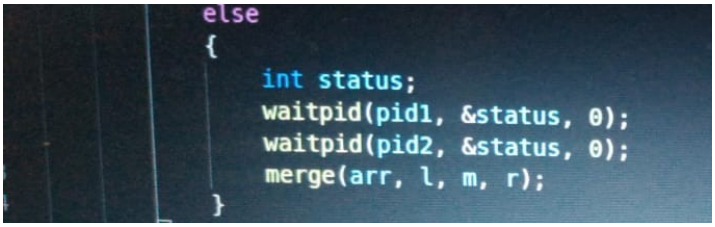
concurrent_merge sort (by forking):

if the size of current part to be sorted is less than 5 then i sorted it using selection sort other wise i created a child for left half, other child to sort the right half and their parent waits for both of them to get sorted. After left and right half got sorted they will exit and their parent merge the two sorted half.

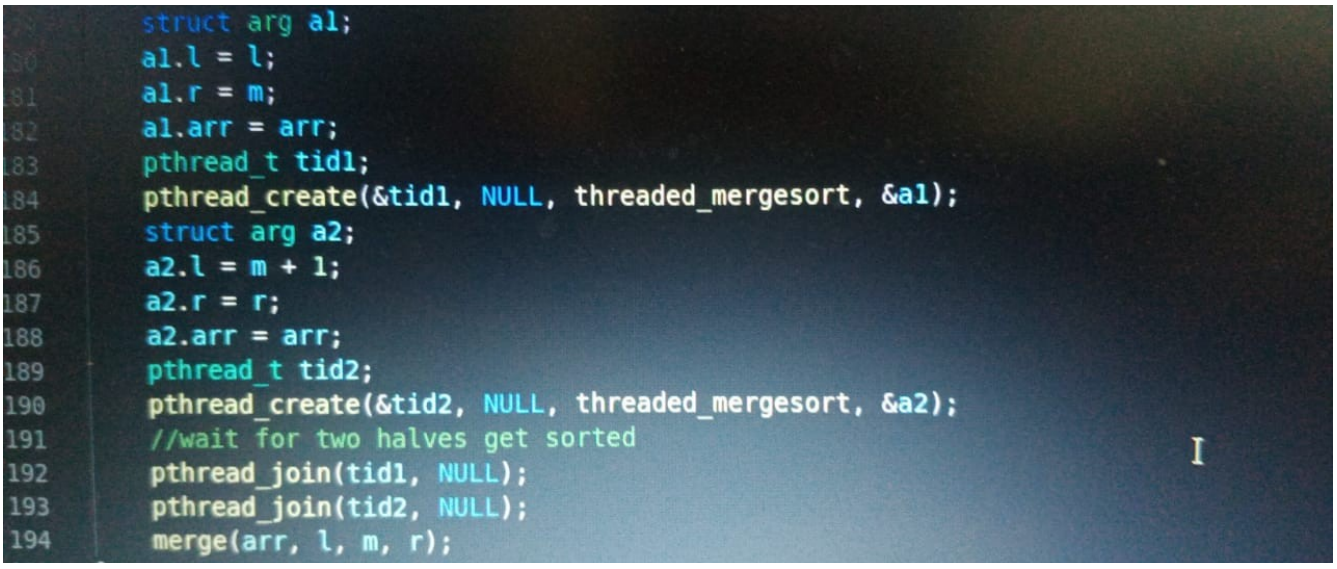


```
if (r - l + 1 <= 4)
{
    selectionsort(arr, l, r);
    return;
}
```



```
else
{
    int status;
    waitpid(pid1, &status, 0);
    waitpid(pid2, &status, 0);
    merge(arr, l, m, r);
}
```

concurrent_merge sort (by threads):
if the size of part of array to be sorted by a thread is less than 5 that thread will sort using select sorta and returns otherwise it creates two threads one for left half, one for right half and the main thread thread waits for two threads to get completed and then merges the result.



```
struct arg a1;
a1.l = l;
a1.r = m;
a1.arr = arr;
pthread_t tid1;
pthread_create(&tid1, NULL, threaded_mergesort, &a1);
struct arg a2;
a2.l = m + 1;
a2.r = r;
a2.arr = arr;
pthread_t tid2;
pthread_create(&tid2, NULL, threaded_mergesort, &a2);
//wait for two halves get sorted
pthread_join(tid1, NULL);
pthread_join(tid2, NULL);
merge(arr, l, m, r);
```

normal merge sort ran faster than concurrent process by threads or by child process.