Operating Systems (327108)

Assignment No. 3

Due Date/Time	28th Dec, 2022 11:59 PM
Files to be submitted	Documentation of each module along with code file.
File Naming	Roll_No_Assign_03.pdf Roll_No_Assign_03.cpp Note: Any assignment that is not according to format will be marked as zero.
Coding Guides	1. Use of proper variable declaration/initialization according to the naming conventions (**camelCase, snake_case, PascalCase**) 2. Use of proper function for each module. Note: Marks will be deducted if not following the above guide line.
Submission Guide line	**1.** Code along with documentations should be submitted on teams by due date/time. **2.** Also update your solution on your **Github** profile. **3.** Also include your **Github** link in the documentation.
Plagiarism	Any kind of plagiarism will result in F grade in course
Weightage	This assignment will be marked on CLOs, also it will be graded for course and lab both. Which means , if you have got 8/10 in this assignment same marks will be reflected for your lab assignment.

This assignment is for testing the understanding of scheduling algorithm you have learn so far, and improve the performance of operating system with some advancement in already available algorithms.

The learning outcome of this assignment is you are able to identify core functions of operating system and how to enhanced the performance with the integration of sorting algorithm.

You have learned merge sort in data structures which sorts an array in **nlogn** time, it is a divide and conquer technique. We can enhance the performance of merge sort using the multithreading.
First of all, you have to check the processor cores of your system, let's suppose your system processor has 4 cores. Now you have to create 4 threads and divide the array among these threads and sort them using merge sort. You have to take size of array and array elements from user.

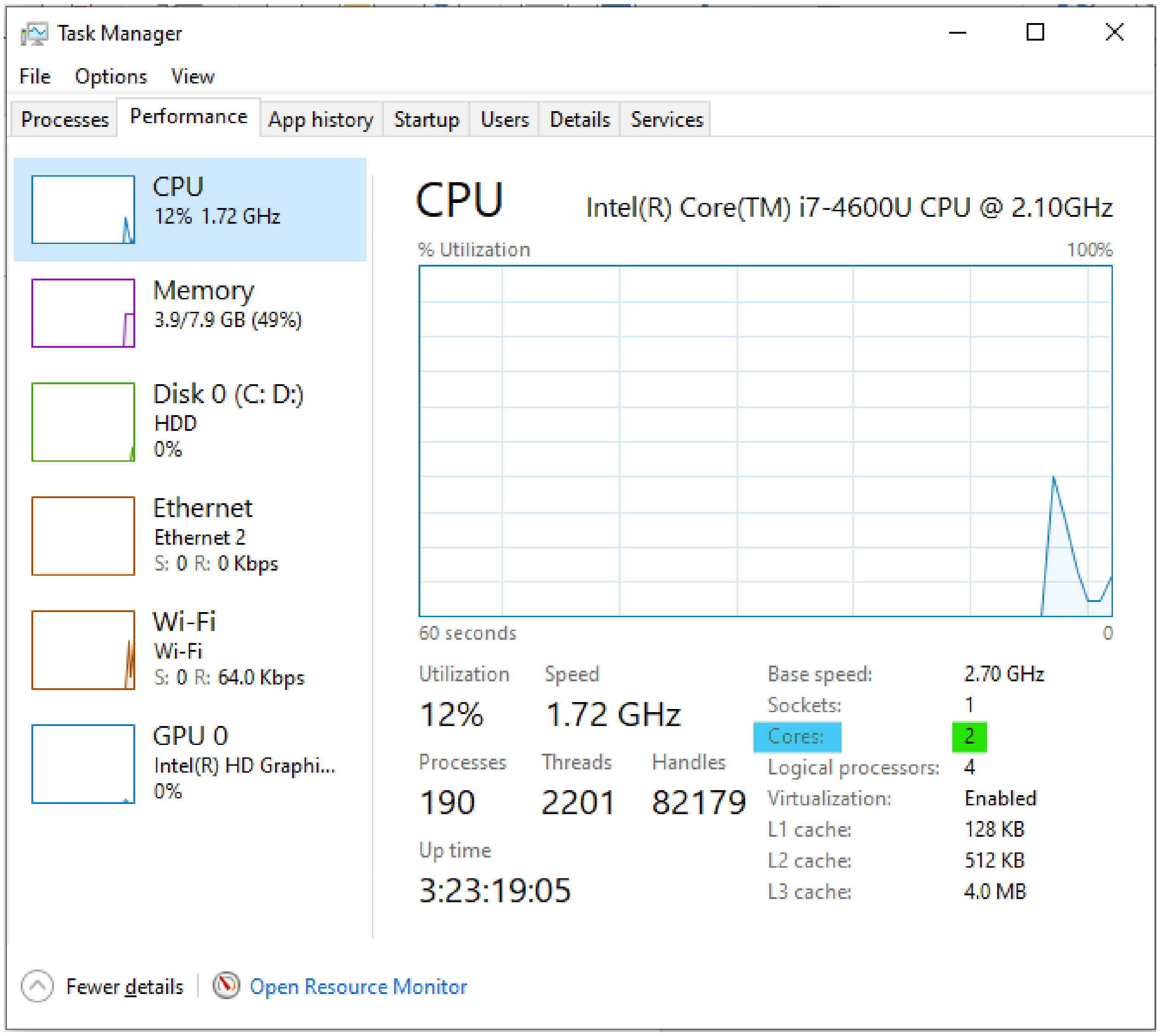
For this question you have to submit three things multithreaded merge sort c/c++ code, screenshot of available cores in your system and also the mac address screenshot of your system. No need to implement merge sort from scratch you can use merge sort code from internet but provide the link of source in the code.

Important: No of threads will be equivalent to no of cores in your system. We will verify no of cores and mac address at the time of demo, if anyone cores and mac address mismatched at the time of demo will be awarded zero marks.

C++ Code:

```
#include <iostream>
#include <pthread.h>
using namespace std;

const int MAX_SIZE = 100;
int array[MAX_SIZE];
int size;

void merge(int arr[], int l, int m, int r)
{
int i, j, k;
int n1 = m - l + 1;
int n2 = r - m;

int L[n1], R[n2];

for (i = 0; i < n1; i++)
L[i] = arr[l + i];
for (j = 0; j < n2; j++)
R[j] = arr[m + 1 + j];

i = 0;
j = 0;
k = l;
while (i < n1 && j < n2)
{
if (L[i] <= R[j])
{
arr[k] = L[i];
i++;
}
else
{
arr[k] = R[j];
j++;
}
k++;
}

while (i < n1)
{
arr[k] = L[i];
i++;
k++;
}

while (j < n2)
{
arr[k] = R[j];
j++;
k++;
}
}

void mergeSort(int arr[], int l, int r)
{
if (l < r) {
int m = l + (r - l) / 2;
mergeSort(arr, l, m);
mergeSort(arr, m + 1, r);

merge(arr, l, m, r);
}
}

void *mergeSortThread(void *args)
{
int *arr = (int *) args;
int l = arr[0];
int r = arr[1];
mergeSort(array, l, r);
return NULL;
}

int main()
{
cout << "Enter the size of the array: ";
cin >> size;
cout << "Enter the elements of the array: ";
for (int i = 0; i < size; i++)
cin >> array[i];

pthread_t thread1, thread2;
int arr1[2] = {0, size/2 - 1};
int arr2[2] = {size/2, size - 1};
pthread_create(&thread1, NULL, mergeSortThread, (void *) arr1);
pthread_create(&thread2, NULL, mergeSortThread, (void *) arr2);
pthread_join(thread1, NULL);
pthread_join(thread2, NULL);

merge(array, 0, size/2 - 1, size - 1);

cout << "Sorted array: ";
for (int i = 0; i < size; i++)
cout << array[i] << " ";
cout << endl;
return 0;
}
```

Screenshot of available cores in my system:

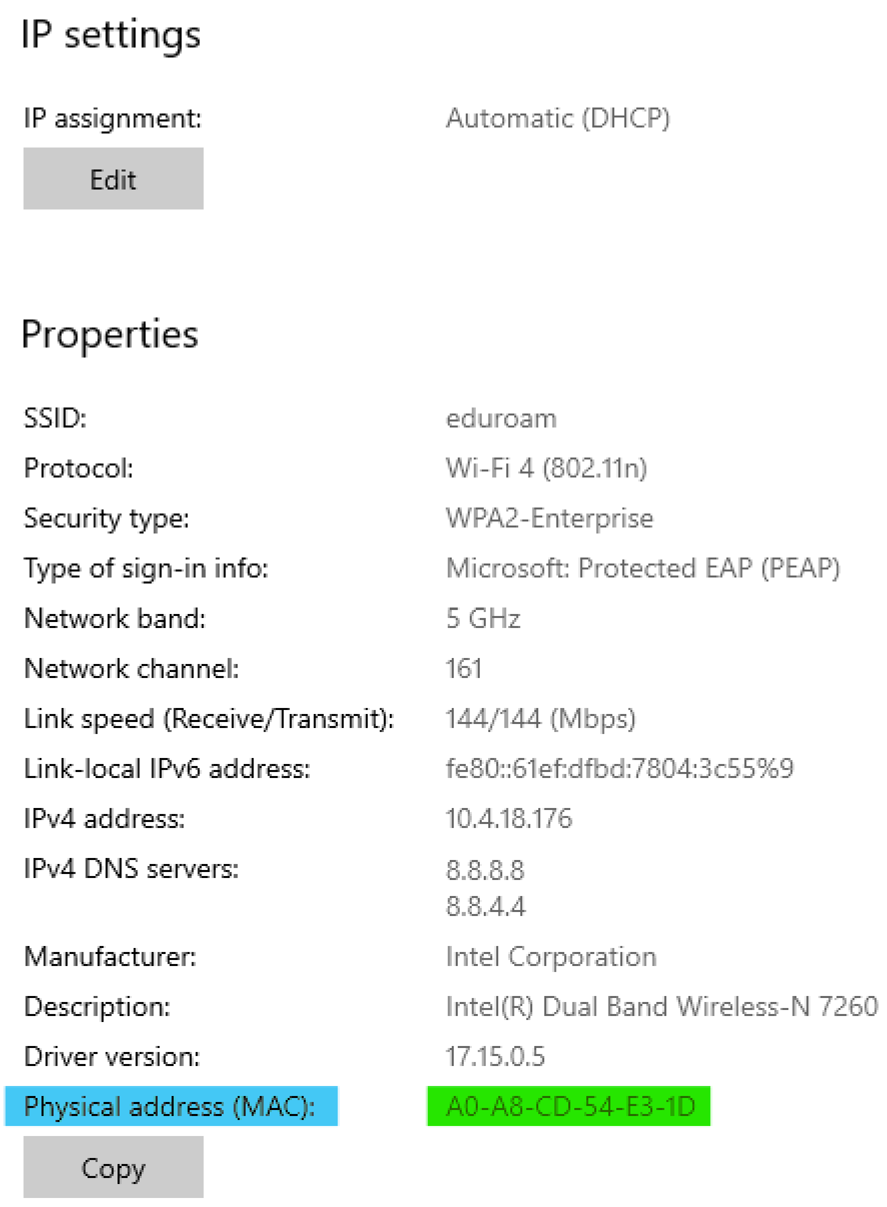


CPU Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz

Utilization	Speed		
12%	1.72 GHz		
Processes	Threads	Handles	
190	2201	82179	
Up time			
3:23:19:05			

Base speed:	2.70 GHz
Sockets:	1
Cores:	2
Logical processors:	4
Virtualization:	Enabled
L1 cache:	128 KB
L2 cache:	512 KB
L3 cache:	4.0 MB

MAC Address:



IP settings

IP assignment:	Automatic (DHCP)

Edit

Properties

SSID:	eduroam
Protocol:	Wi-Fi 4 (802.11n)
Security type:	WPA2-Enterprise
Type of sign-in info:	Microsoft: Protected EAP (PEAP)
Network band:	5 GHz
Network channel:	161
Link speed (Receive/Transmit):	144/144 (Mbps)
Link-local IPv6 address:	fe80::61ef:dfbd:7804:3c55%9
IPv4 address:	10.4.18.176
IPv4 DNS servers:	8.8.8.8 8.8.4.4
Manufacturer:	Intel Corporation
Description:	Intel(R) Dual Band Wireless-N 7260
Driver version:	17.15.0.5
Physical address (MAC):	A0-A8-CD-54-E3-1D

Copy

"THE END"